

Springer

Berlin

Heidelberg

New York

Barcelona

Hong Kong

London

Milan

Paris

Singapore

Tokyo

Panos Antsaklis Wolf Kohn
Michael Lemmon Anil Nerode
Shankar Sastry (Eds.)

Hybrid Systems V



Springer

Series Editors

Gerhard Goos, Karlsruhe University, Germany
Juris Hartmanis, Cornell University, NY, USA
Jan van Leeuwen, Utrecht University, The Netherlands

Volume Editors

Panos Antsaklis
Michael Lemmon
Department of Electrical Engineering, University of Notre Dame
Notre Dame, IN 46556 USA
E-mail: Panos.J.Antsaklis.1@nd.edu
lemmon@maddog.ee.nd.edu

Wolf Kohn
Hynomics Corporation
10632 NE 37th Circle, Building 23, Kirkland, WA 98033, USA
E-mail: wk@hybrithms.com

Anil Nerode
Department of Mathematics, Cornell University
217, White Hall, Ithaca, NY 14853, USA
E-mail: anil@math.cornell.edu

Shankar Sastry
Electronic Research Laboratory, University of California at Berkeley
253 Cory Hall, Berkeley, CA 94720, USA
E-mail: sastry@eecs.berkeley.edu

Cataloging-in-Publication data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Hybrid systems. - Berlin ; Heidelberg ; New York ; Barcelona ;
Budapest ; Hong Kong ; London ; Milan ; Paris ; Santa Clara ;
Singapore ; Tokyo : Springer
Literaturangaben
5. / Panos Antsaklis ... (ed.) 1999
(Lecture notes in computer science ; Vol. 1567)
ISBN 3-540-65643-X

CR Subject Classification (1998): C.1.m, C.3, D.2.1, F.3.1, F.1.2, J.2

ISSN 0302-9743

ISBN 3-540-65643-X Springer-Verlag Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag. Violations are liable for prosecution under the German Copyright Law.

© Springer-Verlag Berlin Heidelberg 1999
Printed in Germany

Typesetting: Camera-ready by author
SPIN 10702989 06/3142 - 5 4 3 2 1 0 Printed on acid-free paper

Preface

Hybrid systems are interacting networks of digital and continuous systems. Hybrid systems arise throughout business and industry in areas such as interactive distributed simulation, traffic control, plant process control, military command and control, aircraft and robot design, and path planning. Three of the fundamental problems that hybrid systems theory should address are: How to model physical and information systems as hybrid systems; how to verify that their behavior satisfies program or performance specifications; and how to extract from performance specifications for a network of physical systems and their simulation models digital control programs which will force the network to obey its performance specification. This rapidly developing area is at the interface of control, engineering and computer science. Methods under development are extensions of those from diverse areas such as program verification, concurrent and distributed processes, logic programming, logics of programs, discrete event simulation, calculus of variations, optimization, differential geometry, Lie algebras, automata theory, dynamical systems, etc.

When the first LNCS volume *Hybrid Systems* was published in 1993, the effect was to focus the attention of researchers worldwide on developing theory and engineering tools applicable to hybrid systems in which continuous processes interact with digital programs in real time. At the time of publication of this fifth volume, there is general agreement that this is an important area in which mathematics, control engineering, and computer science can be fruitfully combined. There are now hybrid system sections in many engineering and computer science international meetings, hybrid systems research groups in many universities and industrial laboratories, and also other excellent series of hybrid systems conferences.

The impetus for this volume was the Fifth International Hybrid Systems Workshop held in Notre Dame, Ind., USA, September 11-13, 1997. Previous Hybrid Systems Workshops have taken place at MSI/Cornell (June 10-12, 1991), the Technical University Lyngby, Denmark (October 19-21, 1992), MSI/Cornell (October 28-30, 1994), DIMACS/SYCON Rutgers (October 22-25, 1995), and MSI/Cornell (October 12-14, 1996). The four volumes arising from these workshops were: *Hybrid Systems*, Springer-Verlag LNCS **736** (1991 and 1992 workshops); *Hybrid Systems II*, Springer-Verlag LNCS **999** (1994 workshop); *Hybrid Systems III*, Springer-Verlag LNCS **1066** (1995 workshop); and *Hybrid Systems IV*, Springer-Verlag LNCS **1273** (1996 workshop). All papers were fully refereed and selected from those submitted in a call subsequent to the Fifth International Hybrid Systems Workshop.

Here are brief summaries of the papers which have been included.

- Chen and Hanisch present a method for synthesizing hybrid feedback policies based on predicate invariance over the hybrid state space of the system.
- Chutinan and Krogh develop the machinery for constructing approximating automata for continuous systems where the continuous dynamics are

defined by convex polytopes in the space of derivatives of continuous space trajectories.

- Davoren proposes the propositional modal $\mu\otimes$ calculus – which subsumes most known temporal and modal propositional logics – as a broad logical framework for the formal analysis and verification of hybrid systems. Over transition system models equipped with a topological or metric structure, the logic can express continuity properties of transition relations and metric tolerance properties such as “being within distance ϵ ” of a set of states, thus allowing formal verification of robustness and stability properties of hybrid dynamical systems.
- De Schutter and De Moor extend the Extended Linear Complementary Problem algorithms previously used for discrete event systems to analyze some classes of hybrid systems. Their case study is of a traffic-light controlled intersection.
- Ferreira and Krogh present the results of simulations based on a neural network model of controller scheduling for real-time switched systems. The switching strategy selects the current controller based on neural network estimates of the future system performance for each controller.
- Gao and Xu model fault diagnosis and isolation as a hybrid system and express system specifications in the duration calculus, a dense time temporal logic. An extension of program logic is used as a framework for formal verification.
- Kohn, Nerode and Rummel introduce models for multiple sensor fusion of data and for synchronization of sensors by a Noether algorithm in the calculus of variations as well as a multiple agent hybrid sensor architecture for such problems.
- Koutsoukas and Antsaklis use a class of timed Petri nets for supervisory control of hybrid systems. When the continuous dynamics are described by first order integrators, this is a linear programming problem.
- Kowalewski, Stursberg, Fritz, Graf, Hoffman, Preußig, Remelhe, Simon and Treseler explain and compare eight computer tools – six simulation packages and two verification tools – with respect to validation of logic control programs for continuous processes, with a two tanks problem as their benchmark example.
- Lafferriere, Pappas and Sastry use recent results in the model theory of first-order structures over the real numbers to establish the existence of finite bisimulation quotients for certain classes of planar hybrid systems, thus proving the decidability of verification problems for such systems.
- Lemmon and Bett build on recent results on multiple agent linear control for systems satisfying a bounded amplitude performance constraint, to give a method of extracting a timed automaton as a logical model of specified switched system behavior.
- Mosterman, Zhao and Biswas give semantics and simulation algorithms for a class of dynamical systems operating in so-called “sliding regimes.” With time and space scale abstractions, these become hybrid systems with chattering.

- Nadjm-Tehrani gives a model of time deterministic hybrid systems in which transitions can have different delays and gives a transition system semantics closed under various operations, including parallel composition.
- Neller transforms initialized bounded-time safety problems for hybrid systems into global optimization problems starting with an initial safety estimate, and compares the results with simulated annealing and multi-level single linkage methods.
- Parisini and Saccone analyze mode switching control policies for switched systems as two-level hybrid control schemes; they prove an invariance result for such schemes and report on simulation results.
- Raisch, Klein, O’Young, Meder and Itigin give a methodology for approximating continuous plant models by non-deterministic discrete automata and synthesizing discrete supervisory control for the approximation. The methodology is illustrated by two examples from process control.
- Rönkko and Ravn extend the action system framework for distributed and reactive systems to hybrid systems with differential equation actions and evolution guards; an extended notion of parallel composition for hybrid action systems is also developed.
- Seibel, Farines and Cury describe a methodology for the design of flight plans for rotary-wing unmanned aerial vehicles based on formal verification using linear hybrid automata.
- Skafadis, Evans, Mareels and Nerode present a solution to certain problems in mode switching controller design for stochastic dynamical systems with quadratic cost design, using dynamic programming to extract the desired control.
- Wang, Khargonekar and Beydoun investigate robust control of hybrid systems in the presence of modeling errors and structural uncertainties. For a class of non-linear hybrid systems, they develop a design methodology for hybrid state feedback based on system performance and prove robust stability results.
- Wong-Toi reduces problems of initialized rectangular automata to linear hybrid automata, yielding semialgebraic algorithms for analyzing slope parameters. Automated analysis is used to extract bounds on independent clock drifts in an audio control protocol. The emptiness problem for slope-parametric rectangular automata is proven undecidable.
- Yu and Chen present a control framework for interval temporal systems, using duration calculus to express properties of open- and closed-loop behavior.
- Zhao, Loh and May describe their phase-space nonlinear control toolbox for synthesizing and evaluating control laws for a wide class of nonlinear systems. The maglev project is used as a testbed.

Acknowledgments: We thank the following organizers of HS97: Panos Antsaklis, General Chair and organizing committee; Michael Lemmon, Program Chair; Wolf Kohn, organizing committee; Anil Nerode, organizing committee;

Shankar Shastry, organizing committee, and the remainder of the program committee: R. Alur, M. Branicky, B. Egardt, T. Henzinger, B. Krogh, J. Lygeros, O. Maler, A.S. Morse, S. Nadjm-Tehrani, J. Raisch, H. Schumacher, and J. Sifakis. Sponsors were the National Science Foundation, the MURI “An Integrated Approach to Intelligent Systems”, DAAH04-96-1-0341 administered by the Army Research Office, the Center for Foundations of Intelligent Systems (Cornell), and the University of Notre Dame. The IEEE Control Systems Society was the technical cosponsor. We are also deeply indebted to J. Davoren and A. Walz for technical editing, manuscript preparation and the administration associated with the volume.

December 1998

Panos Antsaklis, Notre Dame, IN
Wolf Kohn, Bellevue, WA
Michael Lemmon, Notre Dame, IN
Anil Nerode, Ithaca, NY
Shankar Sastry, Berkeley, CA

Table of Contents

Control Synthesis of Hybrid Systems Based on Predicate Invariance	1
<i>H. Chen and H.-M. Hanisch</i>	
Computing Approximating Automata for a Class of Linear Hybrid Systems	16
<i>A. Chutinan and B.H. Krogh</i>	
On Hybrid Systems and the Modal μ -calculus	38
<i>J.M. Davoren</i>	
The Extended Linear Complementarity Problem and the Modeling and Analysis of Hybrid Systems	70
<i>B. De Schutter and B. De Moor</i>	
Controller Scheduling Using Neural Networks: Implementation and Experimental Results	86
<i>E.D. Ferreira and B.H. Krogh</i>	
Rigorous Design of a Fault Diagnosis and Isolation Algorithm	100
<i>J. Gao and Q. Xu</i>	
Scalable Data and Sensor Fusion via Multiple Agent Hybrid Systems	122
<i>W. Kohn, A. Nerode and J.B. Remmel</i>	
Hybrid Control Systems Using Timed Petri Nets: Supervisory Control Design Based on Invariant Properties	142
<i>X.D. Koutsoukos and P.J. Antsaklis</i>	
A Case Study in Tool-Aided Analysis of Discretely Controlled Continuous Systems: The Two Tanks Problem	163
<i>S. Kowalewski, O. Stursberg, M. Fritz, H. Graf, I. Hoffmann, J. Preußig, M. Remelhe, S. Simon and H. Treseler</i>	
Hybrid Systems with Finite Bisimulations	186
<i>G. Lafferriere, G.J. Pappas and S. Sastry</i>	
Extracting Stable Timed Automata from Switched Model Reference Systems	204
<i>M. Lemmon and C.J. Bett</i>	
Sliding Mode Model Semantics and Simulation for Hybrid Systems	218
<i>P.J. Mosterman, F. Zhao and G. Biswas</i>	
Time-Deterministic Hybrid Transition Systems	238
<i>S. Nadjm-Tehrani</i>	
Heuristic Optimization and Dynamical System Safety Verification	251
<i>T.W. Neller</i>	
A Hybrid Receding-Horizon Control Scheme for Nonlinear Discrete-Time Systems	262
<i>T. Parisini and S. Sacone</i>	
Approximating Automata and Discrete Control for Continuous Systems – Two Examples from Process Control	279
<i>J. Raisch, E. Klein, S. O’Young, C. Meder and A. Itigin</i>	

Action Systems with Continuous Behaviour	304
<i>M. Rönkkö and A.P. Ravn</i>	
Towards Using Hybrid Automata for the Mission Planning of Unmanned Aerial Vehicles	324
<i>C.W. Seibel, J.-M. Farines and J.E.R. Cury</i>	
Optimal Controller Switching for Stochastic Systems	341
<i>E. Skafidas, R.J. Evans, I.M.Y. Mareels and A. Nerode</i>	
Robust Control of Hybrid Systems: Performance Guided Strategies	356
<i>L.Y. Wang, P.P. Khargonekar and A. Beydoun</i>	
Analysis of Slope-Parametric Rectangular Automata	390
<i>H. Wong-Toi</i>	
Control of Interval Temporal Systems	414
<i>X. Yu and Z. Chen</i>	
Phase-Space Nonlinear Control Toolbox: The Maglev Experience	429
<i>F. Zhao, S.C. Loh and J.A. May</i>	
Author Index	445

Control Synthesis of Hybrid Systems Based on Predicate Invariance^{*}

Haoxun Chen²¹ and Hans-Michael Hanisch¹

¹ University of Magdeburg, Institute for Automation Technology
PO Box 4120, D-39016 Magdeburg, Germany
{hxchen, hami}@hamlet.et.uni-magdeburg.de

² Systems Engineering Institute, Xi'an Jiaotong University
Xi'an 710049, P.R. China

Abstract. A hybrid system is modeled by a hybrid automaton and its control specification is given by a predicate defined on the hybrid state space of the system. The problem is to find a hybrid state feedback policy that enforces the closed-loop system satisfying the specification. In this paper, we present a control synthesis method for the problem based on predicate invariance.

1 Introduction

Hybrid systems consist of a continuous time system and a discrete event system that interact and influence each other. Many artificial systems belong to the hybrid system family, instances of these systems include logic-based switching control systems, intelligent vehicle/highway systems and chemical batch processes. A growing need for modeling, analysis and design of hybrid systems in practice has attracted many researchers' attention in control and computer science communities, and a considerable effort has been made to develop theoretical frameworks and models for such systems.

Most work on hybrid systems in past was focused on modeling and analysis (verification), relatively little on control synthesis. Alur *et. al.* [ACH95] proposed a symbolic model-checking method for verification of linear hybrid systems based on reachability analysis over the infinite state space by iteratively computing sets of states. With this, a software tool HyTech was developed for analysis of the systems [HHH95]. The method was then extended to slope-parametric rectangular automata [H97]. There also have been papers dealing with analysis of nonlinear hybrid systems, but most of them addressed approximation-based methods [CK97] [RY97]. For control synthesis, Tittus *et. al.* [TE94] proposed a method for integrator systems (a class of linear hybrid systems) based on iteratively computing extended jump sets, like the model-checking method of [ACH95]. A specific control synthesis problem of hybrid systems was formulated as an optimal control problem with both discrete and continuous variables in Branicky *et.*

^{*} This research was supported by Alexander von Humboldt Foundation

al. [BM95], but generally the optimal control problem is hard to be solved. Stiver *et. al.* [SAL95] proposed a hybrid control system synthesis approach based on natural invariants, however it requires quantizing and bounding the state space so that the approach can be implemented by a computer algorithm. In addition, some work on hybrid systems was focused on stability issue (see Branicky *et. al.* [B94] and Pettersson *et. al.* [P96]).

For general hybrid systems, it seems computationally intractable (undecidable) for analysis and control synthesis. However, it may be possible to provide some theoretical results with which and with specific structures of real problems the problems can be successfully solved in finite computation time. In this paper, we present a method for control synthesis of hybrid systems based on predicate invariance. The hybrid system we consider is assumed to be described by a hybrid automaton similar to that proposed by Tittus *et. al.* [TE94], and the specification for the system (or the control objective of the system) is given by a predicate defined on the state set of the hybrid system (each state in the set consists of a discrete state and a continuous state), which is usually characterized by a collection of linear inequalities.

The aim is to find a hybrid control policy that makes the closed loop system satisfying the specification. We first introduce control-invariant predicates for the hybrid system and then prove that, for a given predicate, the necessary and sufficient condition for the existence of a control policy that verifies the predicate is that it is control-invariant. With this result, we propose a fixpoint algorithm for computing the extremal control-invariant predicate. On the basis of this, a desired control policy for the hybrid system may be formally synthesized. A control synthesis problem of an evaporation process in a benchmark batch plant is given to illustrate the algorithm.

2 Hybrid Plant Model

An open loop hybrid plant is described by a seven-tuple

$$HP = (Q, \Sigma_h, \delta, X, f, g, \gamma_p) \quad (1)$$

where the components can be divided into three groups:

a) Discrete event part

Q is a finite set of discrete states q

Σ_h is the set of hybrid plant events σ . This set $\Sigma_h = \Sigma_p \cup \Sigma_c$ is partitioned into the set of physical events Σ_p and the set of control events Σ_c . The events in Σ_p occur as a result of the continuous state evolution and are uncontrollable, while the control events in Σ_c are external inputs and are controllable.

$$\delta : Q \times \Sigma_h \rightarrow Q \quad (2)$$

is the (discrete) state transition function for the hybrid system .

b) Continuous part

$X \subseteq R^n$ is a subset of the n -dimensional real valued state space, and $x(t) \in X$ is the corresponding state vector.

The function $f : X \times Q \rightarrow X$ is the vector field of the state vector x , that is

$$\dot{x}(t) = f(x(t), q) \quad (3)$$

where $q \in Q$ is the present discrete state of the hybrid system.

The function $g : X \times Q \times \Sigma_h \rightarrow X$ specifies discontinuous jumps in the state vector at discrete state transitions, i.e.,

$$x(t^+) = g(x(t^-), q, \sigma) \quad (4)$$

where t is the time when the event occurs at state q .

c) Physical event generator

$$\gamma_p : Q \times X \rightarrow 2^{\Sigma_p} \quad (5)$$

The function γ_p implicitly defines a set of regions in the continuous state space for each discrete state, i.e., $R_\sigma^q = \{x \in X | \sigma \in \gamma_p(q, x)\}$, $\sigma \in \Sigma_p$. When the continuous state $x(t)$ enters one of the regions at discrete state q , i.e., $x(t) \in R_\sigma^q$ for some σ , the corresponding physical event σ will occur immediately if no conflict with other events is existed. It should be noted that there may exist more than one events σ such that $x(t) \in R_\sigma^q$. In this case, only one event of them can occur immediately, i.e., they are conflict events at the discrete state.

3 State Feedback Controller

The state feedback controller for the hybrid plant we consider is defined as

$$\gamma_c : Q \times X \rightarrow 2^{\Sigma_c} \quad (6)$$

It can be regarded as an event generator for control events. The closed loop system $CHP = (HP, \gamma_c)$ with the state feedback control structure is shown in Fig. 1. The behaviour of the closed loop system is as follows: When the system is at a discrete state q , its continuous state evolves according to differential equation (3). A physical event (resp. a control event) occurs as soon as the continuous state enters one of the regions specified by physical event generator γ_p (resp. control event generator γ_c). A discontinuous jump in the continuous state may occur with the occurrence of the event, which is governed by the function (4).

Given an open-loop plant model and a specification for the desired closed-loop behavior, the aim is to obtain such a control event generator which satisfies the specification for the closed loop system. In this paper, we assume that the specification is given by a predicate (a state subset) $P : Q \times X \rightarrow \{0, 1\}$ defined on the hybrid state space $Q \times X$. The predicate is usually defined through a set

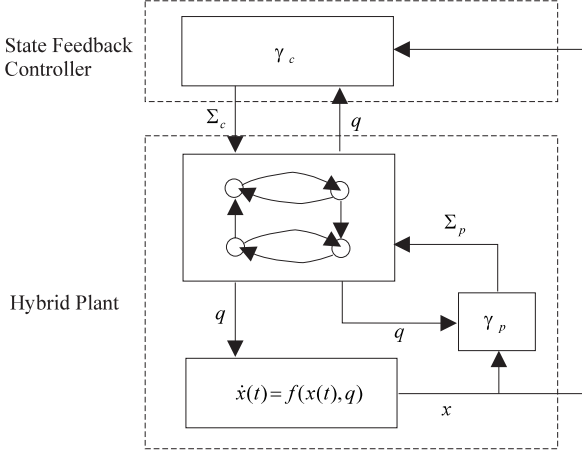


Fig. 1. Closed loop system

of mode invariants and/or system invariants [TE94], where the mode invariants are specified by labeled inequalities

$$\rho_{inv}(q) : g_q(x) \leq 0 \quad (7)$$

In the following, the expressions $P(q, x) = 1$ (resp. $P(q, x) = 0$) and $(q, x) \in P$ (resp. $(q, x) \notin P$) are thought as equivalent ones, and we say (q, x) satisfies (resp. does not satisfy) predicate P if $P(q, x) = 1$ (resp. $P(q, x) = 0$).

4 Control-invariant Predicate

The following notations will be used in the sequel

$P_q = \{x | (q, x) \in P\}$: the continuous state set at discrete state q , given by the predicate P .

P_q^0 and ∂P_q : the internal point set and the boundary point set of P_q respectively.

$\Sigma_{q,x}$: the set of events that are plant-enabled at state (q, x) , i.e., $\Sigma_{q,x} = \{\sigma \in \Sigma_p | \delta(q, \sigma)! \wedge \sigma \in \gamma_p(q, x)\} \cup \{\sigma \in \Sigma_c | \delta(q, \sigma)!\}$, where $\delta(q, \sigma)!$ means that $\delta(q, \sigma)$ is defined.

$\tilde{x}(t, x, q)$: the continuous state trajectory of differential equation (3) with initial state $\tilde{x}(0, x, q) = x$. We use symbol $\tilde{x}$ to represent a state trajectory and x to represent a point at the state trajectory.

$FD(x, P_q)$: the set of feasible directions at x for set P_q . Here, a vector d is said to be a feasible direction at x for set P_q , if and only if $\exists \varepsilon > 0, x + rd \in P_q^0, \forall 0 < r < \varepsilon$.

$P_q^f = \{x | x \in P_q^0 \vee \{x \in \partial P_q \wedge f(x, q) \in FD(x, P_q)\}\}$ is the set of all continuous states from which the continuous state of the system will remain in set P_q for a sufficiently short period at least.

With these notations, we define the predicate:

$$D_\sigma(q, x) = \begin{cases} 1 & \delta(q, \sigma)! \wedge \sigma \in \gamma_p(q, x) \\ 0 & \text{otherwise} \end{cases}, \quad \sigma \in \Sigma_p \quad (8)$$

D_σ is the predicate identifying the set of states at which event σ is plant-enabled.

We define the predicate transformations:

$$wp_\sigma(P)(q, x) = \begin{cases} 1 & \delta(q, \sigma)! \wedge \sigma \in \gamma_p(q, x) \wedge q' = \delta(q, \sigma) \wedge g(x, q, \sigma) \in P_{q'}^f \\ 0 & \text{otherwise} \end{cases}, \quad \sigma \in \Sigma_p \quad (9)$$

$$wp_\sigma(P)(q, x) = \begin{cases} 1 & \delta(q, \sigma)! \wedge q' = \delta(q, \sigma) \wedge g(x, q, \sigma) \in P_{q'}^f \\ 0 & \text{otherwise} \end{cases}, \quad \sigma \in \Sigma_c \quad (10)$$

$wp_\sigma(P)$ is called the *weakest precondition* of P under σ . $wp_\sigma(P)(q, x) = 1$ if and only if 1) σ is plant-enabled at state (q, x) and the occurrence of event σ leads to a new state which satisfies predicate P . 2) From the new state, the state of the system will keep satisfying P for a sufficiently short period at least. Note that the enabling condition for physical event σ is $\delta(q, \sigma)! \wedge \sigma \in \gamma_p(q, x)$ but $\delta(q, \sigma)!$ for control event σ . This is why the formulation of $wp_\sigma(P)$ for physical events and that for control events are little different.

$$wlp_\sigma(P) = wp_\sigma(P) \vee \sim D_\sigma \quad (11)$$

$wlp_\sigma(P)$ is called the *weakest liberal precondition* of P under σ . $wlp_\sigma(P)(q, x) = 1$ if and only if σ is not plant-enabled at state (q, x) or from the new state resulted from the occurrence of σ at state (q, x) , the state of the system will keep satisfying P for a sufficiently short period at least.

$$wp_{clock}(P)(q, x) = \begin{cases} 1 & \text{there is } T \geq 0 \text{ and } \sigma \in \Sigma_{q, \tilde{x}(T, x, q)} \\ & \text{such that } P(q, \tilde{x}(t, x, q)) = 1 \forall t \in [0, T] \\ & \text{and } wp_\sigma(P)(q, \tilde{x}(T, x, q)) = 1 \\ 0 & \text{otherwise} \end{cases} \quad (12)$$

$wp_{clock}(P)$ is called the *weakest precondition* of P under *clock (time advance)*. $wp_{clock}(P)(q, x) = 1$ means that at state (q, x) the system can be switched to a new mode (a new discrete state) so that the resulted new state satisfies P for a sufficiently short period at least, or starting from state (q, x) the system can run for a period of time, during the period the state of the system keep satisfying P , and at the end of the period the system can be switched to a new mode with the resulted new state satisfying P for a sufficiently short period at least. More formally, $wp_{clock}(P)(q, x)$ can be rewritten as

$$wp_{clock}(P)(q, x) = \bigvee_{T=0}^{\infty} \{ \bigvee_{\sigma \in \Sigma_{q, \tilde{x}(T, x, q)}} wp_\sigma(P)(q, \tilde{x}(T, x, q)) \wedge \bigwedge_{t=0}^T P(q, \tilde{x}(t, x, q)) \} \quad (13)$$

To exclude the physically unrealistic possibility that the time advance might be preempted indefinitely by repeated occurrence of a set of events without time elapse, we impose a technical condition that any two events in the hybrid system can not occur successively without time elapse (*Assumption A*).

Definition 1. A predicate P is said to be control-invariant if

$$P \leq \{\wedge_{\sigma \in \Sigma_p} wlp_{\sigma}(P)\} \wedge wp_{clock}(P) \quad (14)$$

From the definition, if a predicate P is control-invariant, then for any $(q, x) \in P$, we have: 1) if there is a physical event σ that is enabled and fired at state (q, x) , the resulted new state will satisfy P as well. 2) at state (q, x) the system can be switched to a new mode (a new discrete state) so that the resulted new state satisfies P , or starting from state (q, x) the system can run for a period of time, during the period the state of the system keep satisfying P , and at the end of the period the system can be switched to a new with the resulted new state satisfying P for a sufficiently short period at least.

The main result of this paper is as follows

Theorem 1. Given a predicate P , there is a state feedback controller of the form (6) such that for the closed loop system, P remains invariantly true whenever it is initially satisfied, if and only if P is control-invariant.

Proof:

IF. Let $\Gamma_c(q, x) = \{\sigma \in \Sigma_c | \delta(q, \sigma)! \wedge q' = \delta(q, \sigma) \wedge g(x, q, \sigma) \in P_{q'}^f\}$,

$\Gamma_p(q, x) = \{\sigma \in \Sigma_p | \delta(q, \sigma)! \wedge \sigma \in \gamma_p(q, x) \wedge q' = \delta(q, \sigma) \wedge g(x, q, \sigma) \in P_{q'}^f\}$,

we first prove $\Gamma_c(q, x) \cup \Gamma_p(q, x) \neq \emptyset$ for any $x \in P_q \setminus P_q^f$ (*Property A*).

From control-invariance of P , we have $(q, x) \in wp_{clock}(P)$. Since $x \in P_q \setminus P_q^f$, from the definition of $wp_{clock}(P)$, there is $\sigma \in \Sigma_h$ which is plant-enabled so that the resulted new state $(q', g(x, q, \sigma))$ by firing σ at state (q, x) satisfies $g(x, q, \sigma) \in P_{q'}^f$, i.e., $\sigma \in \Gamma_c(q, x) \cup \Gamma_p(q, x)$, thus $\Gamma_c(q, x) \cup \Gamma_p(q, x) \neq \emptyset$.

Define a state feedback controller as follows:

$$\gamma_c : Q \times X \rightarrow 2^{\Sigma_c}$$

$$\gamma_c(q, x) \subseteq \Gamma_c(q, x), \text{ if } x \in P_q^f$$

$$\gamma_c(q, x) \subseteq \Gamma_c(q, x) \text{ and } \gamma_c(q, x) \cup \Gamma_p(q, x) \neq \emptyset, \text{ if } x \in P_q \setminus P_q^f \quad (15)$$

Such a state feedback is existed because of Property A. For the closed loop system $CHP = (HP, \gamma_c)$ and a state $(q, x) \in P$, there are two cases. One is that there is no event $\sigma \in \Sigma_h$ enabled at the state. For this case, $x \in P_q^f$. The system can run for at least a period of time, during the period the state of the system keep satisfying P . The other is that there is an event $\sigma \in \Sigma_h$ enabled at state (q, x) . For this case, $\sigma \in \Gamma_c(q, x)$ or $\sigma \in \Gamma_p(q, x)$, the firing of each of them at state (q, x) will lead to a new state $(q', g(x, q, \sigma))$ with $g(x, q, \sigma) \in P_{q'}^f$, so the new state also satisfies P . Summarily, P remains invariantly true for the closed loop system whenever it is initially satisfied.

ONLY IF. If $P \leq \{\wedge_{\sigma \in \Sigma_p} wlp_\sigma(P)\} \wedge wp_{clock}(P)$ is not true, then we have $(q, x) \in P$, such that $(q, x) \notin \wedge_{\sigma \in \Sigma_p} wlp_\sigma(P)$ or $(q, x) \notin wp_{clock}(P)$. If $(q, x) \notin \wedge_{\sigma \in \Sigma_p} wlp_\sigma(P)$, then there is $\sigma \in \Sigma_p$ that is plant-enabled at state (q, x) such that $g(x, q, \sigma) \notin P_{q'}^f$, $q' = \delta(q, \sigma)$. Since σ is uncontrollable, recalling the assumption A, we assert that the truth of P will be violated during the state evolution of the closed loop system for any state feedback controller. For $(q, x) \notin wp_{clock}(P)$, we can similarly prove the assertion. However, the assertion is contrary to the premise.

From the proof of Theorem 1, if a predicate P is control-invariant, then a state feedback controller (a control event generator) which satisfies the specification can be synthesized according to (15).

To check if a predicate P is control-invariant, we are required to compute $wlp_\sigma(P)$ for each $\sigma \in \Sigma_p$ and $wp_{clock}(P)$. The computation of $wlp_\sigma(P)$ is easy but of $wp_{clock}(P)$ not. However, with some conditions on vector field $f(x, q)$ of the continuous state at each discrete state q , on physical event generator γ_p and on predicate P , it is still possible to compute $wp_{clock}(P)$ efficiently.

For example, suppose that the solution $\tilde{x}(t, x, q)$ of the differential equation (3) can be explicitly obtained and that $f(x, q)$, γ_p and P satisfy the following monotonicity conditions:

- 1) For each (q, x) , $\sigma \in \Sigma_{q, \tilde{x}(t, x, q)}$ for some $t \geq 0$ implies $\sigma \in \Sigma_{q, \tilde{x}(t', x, q)}$ for any $t' \geq t$,
- 2) For each (q, x) , $P(q, x) = 1$ and $P(q, \tilde{x}(t, x, q)) = 1$ for some $t \geq 0$ imply $P(q, \tilde{x}(t', x, q)) = 1$ for any $t' \leq t$.

Then, $wp_{clock}(P)$ can be computed in the following way:

- 1) For each q , $\sigma \in \Sigma_q = \{\sigma \in \Sigma_h \mid \delta(q, \sigma) \neq \emptyset\}$, we compute $t_{q, \sigma}(x) = \min\{t \mid \sigma \in \Sigma_{q, \tilde{x}(t, x, q)}, t \geq 0\}$,
- 2) For each q , we compute $t_{q, P}(x) = \max\{t \mid P(q, \tilde{x}(t', x, q)) = 1, \text{ for any } 0 \leq t' \leq t\}$.
- 3) $wp_{clock}(P)(q, x) = 1$ if $t_{q, \sigma}(x) \leq t_{q, P}(x)$ and there is $t_{q, \sigma}(x) \leq t' \leq t_{q, P}(x)$ such that $wp_{clock}(P)(q, \tilde{x}(t', x, q)) = 1$, otherwise $wp_{clock}(P)(q, x) = 0$.

The applicability of the above procedure depends on whether we can obtain explicit expressions of $t_{q, \sigma}(x)$ and $t_{q, P}(x)$.

5 Extremal Control-invariant Subpredicate

Theorem 1 in last section does not imply that if predicate P is not control-invariant, then we can not find a state feedback controller such that for the closed loop system, P remains invariantly true for some initial state. But in this case, the initial state must be taken from a control-invariant subset (subpredicate) of P . There are probably many such control-invariant subsets. Among them, we pay attention to the ones which are good control-invariant approximations to P .

Let $\mathcal{Q}$ be the set of all predicates defined on $Q \times X$. For $P \in \mathcal{Q}$, we define:

$$CI_{<}(P) = \{P' \mid P' \in \mathcal{Q}, P' \leq P \text{ and } P' \text{ is control-invariant}\}$$

$CI_{<}(P)$ is the set of control-invariant predicates on $Q \times X$ that are stronger than P . It can be easily proved that $CI_{<}(P)$ is nonempty and is closed under arbitrary disjunctions (unions). With this property, we can assert the existence and the uniqueness of the maximal elements of $CI_{<}(P)$. Let $P^\dagger$ denote the maximal element. $P^\dagger$ can be thought as the best control-invariant approximation to P among the predicates which are stronger than P .

We can develop a fixpoint algorithm for the computation of $P^\dagger$. For this purpose, we introduce the map:

$$H : \mathcal{Q} \rightarrow \mathcal{Q}$$

$$H(P') = P' \wedge \{\wedge_{\sigma \in \Sigma_p} wlp_\sigma(P')\} \wedge wp_{clock}(P') \quad (16)$$

and define the sequence of predicates:

$$P_0 = P, P_{k+1} = H(P_k), k = 0, 1, 2, \dots \quad (17)$$

It can be proved that the sequence defined in the above is monotone decreasing and $P^\dagger = \lim_{k \rightarrow \infty} P_k$. With this result, we can compute $P^\dagger$ by the iterative process given by (16) and (17). It should be noted that it is not clear at present if the fixpoint algorithm always terminates in finite steps.

The notion of predicate invariance and the algorithm presented in this paper can be regarded as an extension of these of [RW87] for discrete event systems to hybrid systems, so readers please refer to [RW87] for the proofs of the results in this section.

For a linear hybrid system and a predicate given by a linear formula, it can be proved that the predicates $P_k, k = 0, 1, 2, \dots$ generated during the run of the fixpoint algorithm are definable as unions of convex polyhedra in multidimensional real space, so that a symbolic computation algorithm like the model-checking procedure implemented in HyTech [HHH95] can be designed to automatically find $P^\dagger$ using computer. It is very difficult to implement such a symbolic computation algorithm for general hybrid systems, however, in the next section, we will show by an example that even for nonlinear hybrid systems, it is possible to efficiently compute $P^\dagger$ by using the theoretical results proposed in previous sections and by taking account of specific structure of the system in the example.

6 Example

6.1 Process description

We use a part of a benchmark plant described in [HK94] as an example. The flowsheet is depicted in Fig. 2. B is an evaporator that transforms low concentration salty water to high concentration salty water by means of evaporation, V1 and V2 are the valves for feed and discharge of salty water respectively, pipe P1 is an outlet for steam generated by B (P1 is connected to a condensor), H is the (electrical) heating. There are continuous sensors LIS, TI and QIS measuring the level, the temperature and the concentration of salty water in the evaporator.

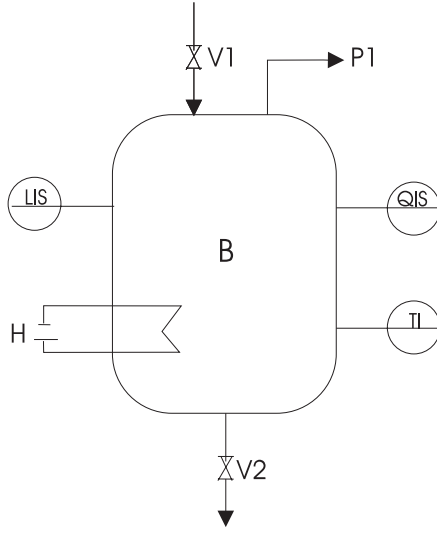


Fig. 2. An evaporation process for dissolved salty water

The recipe of the evaporation process is as follows: An amount of low concentration salty water is first filled into B, and is then heated to evaporate. During the evaporation process, the steam generated by B goes out from B through pipe P1. The process will be terminated when the concentration of the salty water reaches to a given value. The salty water with given concentration is afterwards discharged through valve V2 as product of the process.

For simplicity of modeling, we assume

a) There is an underlying controller that prevents valves V1 and V2 from being opened concurrently, and from being opened while the salty water in B is being heated.

b) The time lag of the heating is negligible.

c) During the filling, discharging and heating, the heat exchange between the salty water and its environment is negligible.

d) The concentration of the salty water in the whole evaporation process is fairly low so that the mass density, the specific heating capacity and the specific evaporation enthalpy of the salty water can be approximated by these of pure water respectively.

e) The flow rate of valve V1 is constant and the flow rate of valve V2 depends on the level of the salty water ($\propto \sqrt{gx}$, where g is the gravity constant and x is the level).

f) B is a standard cylinder

6.2 Hybrid model

We need the following *parameters* to set up our hybrid automaton model of the process (see Fig. 3):

- r_1 : flow rate of valve V1 (l/s), suppose that r_1 is a constant
- r_2 : flow rate of valve V2 (l/s), $r_2 = k_2 \sqrt{gx}$, where k_2 is a constant, x is the level of the salty water, g is the gravity constant
- c : heat capacity of water (kJ/kgC⁰)
- e : evaporation enthalpy of water (kJ/kgC⁰)
- c_0 : concentration of the salty water filling into B through valve V1 (kg/m³)
- c_d : desired concentration of the salty water (kg/m³), $c_d \geq c_0$
- ρ : mass density of water (kg/m³)
- P : power of the heating (kJ/s)
- S : square of the base of B (m²)
- $\alpha_1 = \frac{P}{c\rho S}$, $\alpha_2 = \frac{P}{e\rho S}$
- h : the maximum admissible level of the salty water in B (m)
- l : the minimum admissible level of the salty water in B (m)
- T_0 : temperature of the salty water filling into B through valve V1 (environment temperature, C⁰)
- k : a parameter for natural fall of the temperature (1/s)

Events (transitions) for the process:

- V_1^+ : open valve V1 (control event)
- V_1^- : close valve V1 (control event)
- F: B is full and the valve V1 is closed automatically (physical event)
- H^+ : turn on the heating (control event)
- Eva: the salty water begins to evaporate (physical event)
- D: the desired concentration is reached and the heating is turned off automatically (physical event)
- LB: the minimum level is reached and the heating is turned off automatically (physical event)
- V_2^+ : open valve V2 (control event)
- E: B is empty and the valve V2 is closed automatically (physical event)

Continuous state variables for the process:

- x_1 : level of the salty water in B (m)
- x_2 : temperature of the salty water in B (C⁰)
- x_3 : concentration of the salty water in B (kg/m³)

Discrete states and their corresponding continuous dynamics for the process

- q_0 (initial state): $\dot{x}_1 = 0$, $\dot{x}_2 = 0$, $\dot{x}_3 = 0$ ($x_1(0) = 0$, $x_2(0) = T_0$, $x_3(0) = c_0$)
- q_1 (filling): $\dot{x}_1 = r_1$, $\dot{x}_2 = 0$, $\dot{x}_3 = 0$
- q_2 (ready to be heated): $\dot{x}_1 = 0$, $\dot{x}_2 = 0$, $\dot{x}_3 = 0$
- q_3 (heating): $\dot{x}_1 = 0$, $\dot{x}_2 = \frac{\alpha_1}{x_1}$, $\dot{x}_3 = 0$
- q_4 (evaporating): $\dot{x}_1 = -\alpha_2$, $\dot{x}_2 = 0$, $\dot{x}_3 = \frac{\alpha_2 x_3}{x_1}$

q_5 (ready to be discharged): $\dot{x}_1 = 0, \dot{x}_2 = k(T_0 - x_2), \dot{x}_3 = 0$

q_6 (discharging): $\dot{x}_1 = -k_2\sqrt{gx_1}, \dot{x}_2 = 0, \dot{x}_3 = 0$

Physical event generator:

$q = q_1, x_1 = h$: the salty water in B reaches its maximum level, event F occurs.

$q = q_3, x_2 = 100$: the temperature of the salty water in B reaches 100°C , event Eva occurs.

$q = q_4, x_3 = c_d$: the salty water in B reaches its desired concentration, event D occurs.

$q = q_4, x_1 = l$: the salty water in B reaches its minimum level, event LB occurs.

$q = q_6, x_1 = 0$: the salty water in B is completely discharged, event E occurs.

Discontinuous state jumps:

at transition E of state q_6 :

$$x_2 \rightarrow T_0, x_3 \rightarrow c_0$$

It means that after the salty water in B is completely discharged, the plant returns to its initial state, a new cycle of the production then begins.

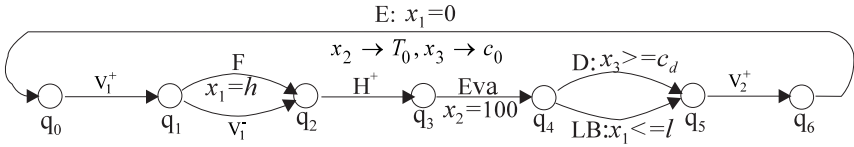


Fig. 3. Hybrid automaton model of the evaporation process

6.3 Specifications for the process

a) system invariants

$$l \leq x_1 \leq h$$

$$0 \leq x_3 \leq c_{max}$$

where the first invariant specifies that the level of the salty water in B must be between its minimum value and maximum value, the second invariant specifies that a maximum concentration (c_{max}) of salty water must not be exceeded, otherwise the salty water in B may boil over (foam may arise and flow to the condensor) during the evaporation phase and may crystallize during the heating phase. We assume $c_{max} \geq c_d$, otherwise it is impossible to produce a batch of desired concentration without violating the system invariant.

b) mode invariants

$$\begin{aligned} c_d - \varepsilon &\leq x_3 \leq c_d + \varepsilon \text{ at } q = q_5 \\ c_d - \varepsilon &\leq x_3 \leq c_d + \varepsilon \text{ at } q = q_6 \end{aligned}$$

where ε is the maximum admissible deviation between the concentration of the produced salty water and the desired concentration. The mode invariants specify that there should be no waste batch produced in the plant.

6.4 Control synthesis based on predicate invariance

According to the hybrid automaton model of the plant, it is clear that the system invariants of the plant have been obeyed, so we only need to consider the mode invariants of the plant. We try to find a maximally permissive state feedback controller that makes the closed loop hybrid system obey all of the mode invariants.

The mode invariants can be described by a predicate (a set equivalent to the predicate):

$$P = \{(q, x) | q \in Q, x \in X \text{ and } c_d - \varepsilon \leq x_3 \leq c_d + \varepsilon \text{ if } q \in \{q_5, q_6\}\}$$

where $Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$, $X = \{(x_1, x_2, x_3) | 0 \leq x_1 \leq h, T_0 \leq x_2 \leq 100, 0 \leq x_3 \leq 1\}$. For simplicity, P will be represented by $P = \{q_5, q_6, c_d - \varepsilon \leq x_3 \leq c_d + \varepsilon\} \vee \{q_0, q_1, q_2, q_3, q_4\}$. Such a convention will be applied to other predicates to be appeared in the following as well. Note that with the convention, set (predicate) $\{q_0, q_1, q_2, q_3, q_4, q_5, q_6\} = \{(q, x) | q \in Q, x \in X\}$.

Note that

$$\Sigma_p = \{F, \text{Eva}, \text{LB}, \text{D}, \text{E}\}, \quad \Sigma_c = \{V_1^+, V_1^-, H^+, V_2^+\}$$

We have:

$$\begin{aligned} wlp_F(P) &= wp_F(P) \vee \sim D_F \\ &= \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\} \\ wlp_{\text{Eva}}(P) &= \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\} \\ wlp_{\text{LB}}(P) &= \{q_4, x_1 \leq l, c_d - \varepsilon \leq x_3 \leq c_d + \varepsilon\} \vee \{q_4, x_1 > l\} \vee \{q_0, q_1, q_2, q_3, q_5, q_6\} \\ wlp_D(P) &= \{q_4, c_d - \varepsilon \leq x_3 \leq c_d + \varepsilon\} \vee \{q_4, x_3 < c_d\} \vee \{q_0, q_1, q_2, q_3, q_5, q_6\} \\ wlp_E(P) &= \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\} \\ wp_{V_1^+}(P) &= \{q_0\} \\ wp_{V_1^-}(P) &= \{q_1\} \\ wp_{H^+}(P) &= \{q_2\} \\ wp_{V_2^+}(P) &= \{q_5, c_d - \varepsilon \leq x_3 \leq c_d + \varepsilon\} \end{aligned}$$

and then

$$\wedge_{\sigma \in \Sigma_p} wlp_{\sigma}(P) = \{q_4, c_d - \varepsilon \leq x_3 \leq c_d + \varepsilon\} \vee \{q_4, x_1 > l, x_3 \leq c_d + \varepsilon\} \\ \vee \{q_0, q_1, q_2, q_3, q_5, q_6\}$$

Note that the solution of the differential equation corresponding to discrete state q_4 (the continuous state trajectory at q_4) is

$$\begin{aligned} \dot{x}_1(t) &= -\alpha_2 t + x_1(0), \\ \dot{x}_2(t) &= x_2(0), \\ \dot{x}_3(t) &= \exp\left(\int_0^t \frac{\alpha_2}{-\alpha_2 \tau + x_1(0)} d\tau\right) x_3(0), \end{aligned}$$

where $(x_1(0), x_2(0), x_3(0))$ is the initial state at q_4 . $x_3(t)$ is monotonously increasing when $-\alpha_2 t + x_1(0) \geq 0$.

At hybrid state (q_4, x) ($x = (x_1, x_2, x_3)$), $wp_{clock}(P)(q_4, x) = 1$ if and only if there is $t \geq 0$ such that for any $t' \in [0, t]$, $-\alpha_2 t' + x_1 \geq l$, $-\alpha_2 t + x_1 = l$, $c_d - \varepsilon \leq \exp(\int_0^t \frac{\alpha_2}{-\alpha_2 \tau + x_1} d\tau) x_3 \leq c_d + \varepsilon$, or $-\alpha_2 t' + x_1 \geq l$, $\exp(\int_0^{t'} \frac{\alpha_2}{-\alpha_2 \tau + x_1} d\tau) x_3 \leq c_d$, $\exp(\int_0^t \frac{\alpha_2}{-\alpha_2 \tau + x_1} d\tau) x_3 = c_d$. Since $\exp(\int_0^t \frac{\alpha_2}{-\alpha_2 \tau + x_1} d\tau)$ is monotonously increasing when $-\alpha_2 t + x_1 \geq 0$, we have: $wp_{clock}(P)(q_4, x) = 1$ if and only if $x_1 \geq l$, $x_3 \leq c_d + \varepsilon$ and there is $t \geq 0$ such that $-\alpha_2 t + x_1 \geq l$, $c_d - \varepsilon \leq \exp(\int_0^t \frac{\alpha_2}{-\alpha_2 \tau + x_1} d\tau) x_3 \leq c_d + \varepsilon$. Obviously, the necessary and sufficient condition for $wp_{clock}(P)(q_4, x) = 1$ is equivalent to $x_1 \geq l$, $x_3 \leq c_d + \varepsilon$, $c_d - \varepsilon \leq \exp(\int_0^{\frac{x_1-l}{\alpha_2}} \frac{\alpha_2}{-\alpha_2 \tau + x_1} d\tau) x_3$ because of monotonicity of $\exp(\int_0^t \frac{\alpha_2}{-\alpha_2 \tau + x_1} d\tau)$.

So, we have:

$$\begin{aligned} wp_{clock}(P) &= \{q_5, q_6, c_d - \varepsilon \leq x_3 \leq c_d + \varepsilon\} \\ &\quad \vee \{q_4, x_1 \geq l, x_3 \leq c_d + \varepsilon, c_d - \varepsilon \leq \exp(\int_0^{\frac{x_1-l}{\alpha_2}} \frac{\alpha_2}{-\alpha_2 \tau + x_1} d\tau) x_3\} \\ &\quad \vee \{q_0, q_1, q_2, q_3\} \\ &= \{q_5, q_6, c_d - \varepsilon \leq x_3 \leq c_d + \varepsilon\} \\ &\quad \vee \{q_4, x_1 \geq l, x_3 \leq c_d + \varepsilon, c_d - \varepsilon \leq \frac{x_1}{l} x_3\} \\ &\quad \vee \{q_0, q_1, q_2, q_3\} \end{aligned}$$

and then

$$\begin{aligned} P_1 &= H(P_0) = H(P) = P \wedge \{\wedge_{\sigma \in \Sigma_p} wlp_{\sigma}(P)\} \wedge wp_{clock}(P) \\ &= \{q_5, q_6, c_d - \varepsilon \leq x_3 \leq c_d + \varepsilon\} \\ &\quad \vee \{q_4, x_1 \geq l, x_3 \leq c_d + \varepsilon, (c_d - \varepsilon)l \leq x_1 x_3\} \\ &\quad \vee \{q_0, q_1, q_2, q_3\} \end{aligned}$$

Similarly, we have:

$$wlp_F(P_1) = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$$

$$wlp_{\text{Eva}}(P_1) = \{q_3, x_2 = 100, x_1 \geq l, x_3 \leq c_d + \varepsilon, (c_d - \varepsilon)l \leq x_1 x_3\} \vee \{q_3, x_2 < 100\} \\ \vee \{q_0, q_1, q_2, q_4, q_5, q_6\}$$

$$wlp_{\text{LB}}(P_1) = \{q_4, x_1 \leq l, x_1 \geq l, x_3 \leq c_d + \varepsilon, (c_d - \varepsilon)l \leq x_1 x_3\} \vee \{q_4, x_1 > l\} \\ \vee \{q_0, q_1, q_2, q_3, q_5, q_6\} \\ = \{q_4, x_1 = l, c_d - \varepsilon \leq x_3 \leq c_d + \varepsilon\} \vee \{q_4, x_1 > l\} \\ \vee \{q_0, q_1, q_2, q_3, q_5, q_6\}$$

$$wlp_{\text{D}}(P_1) = \{q_4, x_3 \geq c_d, x_1 \geq l, x_3 \leq c_d + \varepsilon, (c_d - \varepsilon)l \leq x_1 x_3\} \vee \{q_4, x_3 < c_d\} \\ \vee \{q_0, q_1, q_2, q_3, q_5, q_6\} \\ = \{q_4, x_1 \geq l, c_d \leq x_3 \leq c_d + \varepsilon\} \vee \{q_4, x_3 < c_d\} \\ \vee \{q_0, q_1, q_2, q_3, q_5, q_6\} \\ wlp_{\text{E}}(P_1) = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$$

$$wpclock(P_1) = \{q_5, q_6, c_d - \varepsilon \leq x_3 \leq c_d + \varepsilon\} \\ \vee \{q_4, q_3, x_1 \geq l, x_3 \leq c_d + \varepsilon, (c_d - \varepsilon)l \leq x_1 x_3\} \\ \vee \{q_0, q_1, q_2\}$$

Thus,

$$P_2 = H(P_1) = P_1 \wedge \{\wedge_{\sigma \in \Sigma_p} wlp_{\sigma}(P_1)\} \wedge wpclock(P_1) \\ = \{q_5, q_6, c_d - \varepsilon \leq x_3 \leq c_d + \varepsilon\} \\ \vee \{q_4, q_3, x_1 \geq l, x_3 \leq c_d + \varepsilon, (c_d - \varepsilon)l \leq x_1 x_3\} \\ \vee \{q_0, q_1, q_2\}$$

Similarly, we have:

$$P_3 = \{q_5, q_6, c_d - \varepsilon \leq x_3 \leq c_d + \varepsilon\} \\ \vee \{q_4, q_3, q_2, x_1 \geq l, x_3 \leq c_d + \varepsilon, (c_d - \varepsilon)l \leq x_1 x_3\} \\ \vee \{q_0, q_1\}$$

$$P_4 = \{q_5, q_6, c_d - \varepsilon \leq x_3 \leq c_d + \varepsilon\} \\ \vee \{q_4, q_3, q_2, q_1, x_1 \geq l, x_3 \leq c_d + \varepsilon, (c_d - \varepsilon)l \leq x_1 x_3\} \\ \vee \{q_0\}$$

$$P_5 = P_4$$

so P_5 is the maximal control-invariant subpredicate of P .

With the maximal control-invariant subpredicate, a (maximally permissive) state feedback controller for the hybrid system can be easily synthesized according to (15).

7 Conclusion

In this paper, we provide a theoretical framework for control synthesis of hybrid systems based on predicate invariance. By a benchmark example, we show that it is possible to efficiently synthesize control laws for hybrid systems by using the theoretical results proposed in the paper and by taking account of specific structures of real systems. Further work is required to propose efficient control synthesis methods for some specific subclasses of hybrid systems in the light of the theoretical framework and to propose some approximation methods for symbolic computation of the extremal control-invariant subpredicate for general hybrid systems.

References

- ACH95. R. Alur, C. Coucoubetis, N. Halbwachs, *et. al*, The algorithmic analysis of hybrid systems, *Theoretical Computer Science*, 138:3-34, 1995.
- BM95. M. S. Branicky and S.K. Mitter, Algorithms for optimal hybrid control, *Proc. IEEE Conf. on Decision and Control*, pp. 2661-2666, New Orleans, Dec. 1995.
- B94. M. S. Branicky, Stability of switched and hybrid systems, *Proc. IEEE Conf. on Decision and Control*, pp.3498-3503, Lake Buena Vista, Dec. 1994.
- CK97. J.E.R. Cury, B.H. Krogh, and T. Niinomi, Synthesis of supervisory controllers for hybrid systems based on approximating automata, *IEEE Trans. on Automatic Control*, to appear in the Special Issue on Hybrid Systems, 1997.
- HK94. H.-M. Hanisch and S. Kowalewski, Diskontinuierlicher Misch- und Trennprozess, Fallbeispiel fuer die Modellierung ereignisdiskreter Systeme und den systematischen Entwurf von Steuerungen (*Benchmark process of GMA Working Group 1.8*), 1994.
- HHH95. T. A. Henzinger, P.-H. Ho and Wong-Toi Howard, A user guide to HYTECH, In E. Brinksma, *et. al*. editors, *Tools and Algorithms for the Construction and Analysis of Systems*, Lecture Notes in Computer Science 1019, pages 41-71, Springer-Verlag, 1995.
- H97. Wong-Toi, Howard, Analysis of slope-parametric rectangular automata, *Fifth Int. Workshop on Hybrid Systems*, Sept. 11-13, 1997, Notre Dame, USA: Final program and abstracts, pp. 198-204.
- P96. S. Pettersson, Modelling, control and stability analysis of hybrid systems, *Technical report No. 247L*, Control Eng. Lab, Chalmers University of Technology, Gteborg, Sweden, 1996.
- RY97. J. Raisch and S. D. OYoung, Discrete approximation and supervisory control of continuous systems, *IEEE Trans. on Automatic Control*, to appear in the Special Issue on Hybrid Systems, 1997.
- RW87. P.J. Ramadge and W. M. Wonham, Modular feedback logic for discrete event systems, *SIAM J. Control and Optimization*, Vol. 25, No. 5, pp. 1202-1218, 1987.
- SAL95. J. A. Stiver, P. J. Antsaklis and M. D. Lemmon, Hybrid control system design based on natural invariants, *Proc. of the 34th Conf. on Decision and Control*, New Orleans, LA, Dec. 1995, pp. 1455-1460.
- TE94. M. Tittus and B. Egardt, Control-law synthesis for linear hybrid systems, *Proc. of the 33rd Conf. on Decision and Control*, pp.961-966, Orlando, FL, USA, 1994.

Computing Approximating Automata for a Class of Linear Hybrid Systems

Alongkritt Chutinan and Bruce H. Krogh *

Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213-3890
`ac4c@andrew.cmu.edu, krogh@ece.cmu.edu`

Abstract. Approximating automata are finite-state representations of the sequential input-output behaviors of hybrid systems characterized by threshold events that trigger discrete changes in the continuous dynamic equations. Procedures proposed for constructing approximating automata require forward and backward mappings of sets of continuous state trajectories – mappings which are not available for arbitrary continuous dynamics. This paper develops the foundations for constructing approximating automata automatically for hybrid systems in which the continuous dynamics are defined by convex polytopes in the vector space of the derivatives of the continuous state trajectories. The computations are illustrated for a simple example which also demonstrates the use of approximating automata to solve verification problems that may be intractable using fixed-point computations for linear hybrid automata.

1 Introduction

This paper concerns the generation of purely discrete models (finite automata) for a class of hybrid systems in which the continuous-state trajectories generate events when specified threshold hypersurfaces are reached. The continuous dynamics are selected by a discrete input signal which switches only when threshold events occur. An algorithm was proposed and illustrated in [3] for computing *approximating automata* for such hybrid systems with arbitrary continuous dynamics. The languages of the approximating automata contain all possible sequences of pairs of inputs and threshold events that are valid for the hybrid system. The approximation can be refined iteratively, converging when possible to a finite automaton that generates the exact sequential behaviors of the hybrid system [9]. In [2] we proposed an alternative algorithm for constructing approximating automata for hybrid systems which can be refined more efficiently than the approximating automata in [3].

The purpose of this paper is to identify a set of hybrid systems for which the approximating automata can be constructed automatically. The algorithms for computing approximating automata rely on mappings that characterize the

* This research was supported in part by DARPA contract F33615-97-C-1012.

forward and backward evolution of families of continuous-state trajectories. It is also necessary to compute the set of continuous states from which no events will be generated, called the set of *null event states*. For arbitrary continuous dynamics, there are no computationally effective ways to compute the forward and backward mappings and the set of null event states. In this paper we consider hybrid systems with continuous dynamics in which the derivative of the continuous state vector lies in a convex polytope, similar to the continuous dynamics in so-called linear hybrid automata [1]. For this case, it is possible to carry out all of the computations necessary to generate the approximating automata using standard computations on linear inequalities and convex polytopes. This paper demonstrates how this can be done in the context of the algorithm in [2]. The same computations could be used in the algorithm in [3].

Although many (but not all) of the computations discussed in this paper are similar to computations used in programs such as HyTech [1] for the verification of linear hybrid automata, approximating automata provide a different approach for hybrid systems analysis and synthesis. To perform verification when the logic for selecting the discrete input based on the threshold events is given, an approximating automaton can be generated first. Properties of the hybrid system can then be verified using tools for finite state systems, such as SMV [7] or COSPAN [6]. One approximating automaton may be used to verify several properties of the hybrid system using these faster, more efficient tools. As shown by the example in section 6, approximating automata may make it possible to verify properties for which the fixed-point computations for the linear hybrid automata will not terminate.

Approximating automata can also be used for the synthesis of discrete controllers. For this application, illustrated in [3], the approximating automata represent the sequential behaviors of the hybrid system under all possible choices of the discrete input at each threshold event. This model can be used as the *plant* to synthesize control logic based on the standard discrete event systems theory [11]. Alternatively, an approximating automaton could replace hybrid system dynamics in parts of a large simulation model to enable much faster evaluation of alternative designs. For both controller synthesis and simulation, the approximating automata provide a finite-state models of all possible sequential input-output behaviors of the hybrid dynamics which can be used as subsystems in larger systems of interconnected input-output models, facilitating modular analysis and design based on discrete techniques.

The concept of approximating automata has also been developed by Raisch and O'Young for discrete-time hybrid systems [10]. Their algorithm for constructing approximating automata is similar to the algorithm in [3], but the forward and backward mappings are given immediately by the discrete-time continuous state transition equations. For continuous-time hybrid systems, the approach of [10] would apply to sampled-data implementations where the sampling times are fixed and known a priori. In the continuous-time model considered in this paper, the "sampling times" are generated by the threshold events, which means the forward and backward mappings must be computed by integrating

the state equations. It is precisely this problem, computing the continuous state trajectories, that is addressed in this paper for a particular class of hybrid systems.

The remainder of the paper is organized as follows. The following section describes the class of hybrid systems to be considered. Section 3 defines the mappings of continuous-state trajectories to be used in the construction algorithm, which is presented in section 4. Section 5 presents the details for calculating the mappings and null event sets for the class of linear hybrid systems defined in section 2. A simple example is presented in section 6 to illustrate the computation of approximating automata for verification. The concluding section summarizes the results in this paper and discusses current research directions.

2 A Class of Linear Hybrid Systems

In this paper we consider the class of hybrid systems illustrated by the block diagram in Fig. 1. The signal $u(\cdot)$ is a right-continuous piecewise constant *condition signal* taking on discrete values in the finite set U . This signal is generated as the state of a condition/event (C/E) system whose state transitions are forced by the *event signal* $v(\cdot)$, a pointwise-nonnzero n_v -dimensional signal taking on values in $\{0, 1\}^{n_v}$. The initial state of the C/E system is given as $u(0^-) \in U$. The set of events that drive transitions in the C/E system are the nonzero values of the event signal $v(\cdot)$, that is, values in the *threshold event set* $V = \{0, 1\}^{n_v} - \{0\}^{n_v}$.

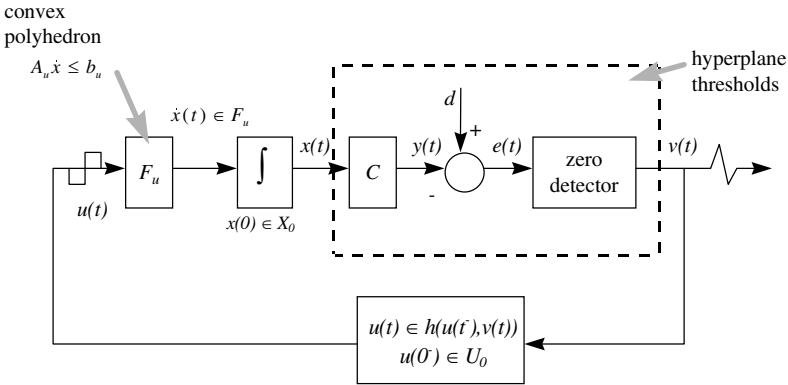


Fig. 1. Block diagram illustrating the class of hybrid systems being considered

The block in Fig. 1 receiving the input signal $u(\cdot)$ is a *signal generator* whose output signal $\dot{x}(\cdot)$ is a piecewise continuous signal with values confined at each instant to be in the set $F_u \subseteq R^{n_x}$, where u is the current value of the input signal $u(\cdot)$ and n_x is the dimension of the continuous state space.

We assume that for each $u \in U$, F_u is a convex polytope defined by the matrix-vector inequality

$$F_u = \{s \in R^{n_x} | A_u s \leq b_u\} \quad (1)$$

where $A_u \in R^{m_u \times n_x}$, $b_u \in R^{m_u}$ and integer m_u are given for each u . We refer to a set of the form (1) as a *derivative constraint set*.

The integrator block in Fig. 1 operates on each component of its input signal to generate the *continuous state trajectory* $x(\cdot)$. X_0 is the set of possible initial continuous states. The continuous state vector is multiplied in the gain block by the *output matrix* $C \in R^{n_v \times n_x}$ creating the continuous output trajectory $y(\cdot)$.

The *threshold events* which constitute the non-zero values of $v(\cdot)$ are generated as follows. Each component of the continuous output vector is subtracted from a component of a constant *threshold vector* $d \in R^{n_v}$, and the resulting signal $e(\cdot) = d - y(\cdot)$ is the input to a *zero detector* which generates the components of the event signal $v(\cdot)$ according to

$$v_i(t) = \begin{cases} 1 & \text{if } e_i(t) = 0 \text{ and } \exists \Delta > 0 \text{ s.t. } e_i(t - \delta) > 0 \text{ for } 0 < \delta < \Delta \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

Note that the zero detector is directional: an event is generated only when a component of $y(t)$ approaches its threshold from below. Note also that a distinct event is generated when more than one threshold is hit at a particular instant. This reflects the fact that when multiple thresholds are encountered simultaneously, the location of the continuous state is known more precisely than when any of the thresholds are encountered individually.

In general, the analysis of the behavior of a hybrid system as described above is virtually intractable if the input signal $u(\cdot)$ can switch at any instant. The scenario of interest is when the input condition signal $u(\cdot)$ changes values only in response to threshold events observed in $v(\cdot)$. This constraint on $u(\cdot)$ is enforced by the assumption that for all u

$$h(u, 0) = \{u\},$$

that is, the C/E system in Fig. 1 allows no transitions when the input event signal is zero. Thus, a signal pair $(u(\cdot), v(\cdot))$ is admissible for the hybrid system only if discontinuities in $u(\cdot)$ occur only when $v(\cdot)$ is non-zero.

To assure $u(\cdot)$ is piecewise constant with only a finite number of switches on any finite interval of time, we assume that $v(\cdot)$ is *nonexplosive* (the term used in the Markov process literature [4]) or *nonzero* (the term popular with computer scientists). That is, we assume that on any finite interval of time $v(\cdot)$ is nonzero a only a finite number of, or zero, times.

Following the development in [8], we define the discrete sequential behavior of the hybrid system described above as the language of the admissible sequences of pairs of discrete inputs and threshold events. In general this language, denoted L_H , contains strings of finite length and sequences of infinite length. Specifically, the sequential behavior corresponding to a signal pair $(u(\cdot), v(\cdot))$ is the sequence

of ordered pairs of the form $(u(t_i^-), v(t_i))$, where the times t_i are the instants at which $v(t) \neq 0$, and $u(t_i^-)$ is the value of the discrete input that lead to the generation of the event $v(t_i)$. If this sequence is finite length, it is terminated by the pair $(u(t_f), v_\perp)$, where t_f is the time of the last threshold event in the signal $v(\cdot)$ and $v_\perp$ is a special symbol introduced to denote this *null threshold event*.

Given a hybrid system, our interest is in creating a finite-state automaton that generates the language L_H . Since such an automaton may not exist, or may be too complex for the purpose at hand, our objective is to generate an approximating automaton A whose language L_A contains L_H . Such an automaton is useful either for verification of system properties or for discrete controller synthesis as described in the introduction. For controller synthesis, the nondeterministic C/E system in Fig. 1 represents the choices available to the supervisor, where some restrictions may be imposed by logic already incorporated into the hybrid system either physically or by lower-level discrete controllers. A supervisor designed to achieve specifications for the sequential behavior for L_A can then be converted to a supervisor which achieves the same specifications for the hybrid system as described in [3]. The remainder of this paper deals with the construction of approximating automata for hybrid systems of the form illustrated in Fig. 1. Throughout the remaining sections the term *hybrid system* refers to this particular type of hybrid system.

3 Mappings of Families of Continuous Trajectories

The sequential behaviors of the hybrid systems are generated by continuous trajectories that evolve between threshold events. Therefore, computing the mappings of families of trajectories from one threshold hyperplane to another is the key issue in the construction of approximating automata. This section introduces notation and definitions to deal with these mappings formally.

Threshold events in a hybrid systems are generated by a set of n_v *threshold hyperplanes* denoted $M_1, \dots, M_{n_v}$. Threshold hyperplane M_i is defined by row i of the output matrix C , and component i of the threshold vector d ; that is,

$$M_i = \{x | c_i^T x = d_i\}$$

Without loss of generality we assume $\|c_i\| = 1$. We denote the *negative*, *positive*, and *non-negative half-spaces* determined by c_i and d_i by $M_i^-(c_i^T x < d_i)$, $M_i^+(c_i^T x > d_i)$, $M_i^{0+}(c_i^T x \geq d_i)$, respectively.

Given a continuous state $x_0 \in R^{n_x}$ and a set $F \subseteq R^{n_x}$, we define the set of admissible continuous-state trajectories starting from x_0 with derivatives lying in F as

$$\mathcal{X}(x_0, F) = \{ x(\cdot) \mid x(0) = x_0, \dot{x}(t) \in F \text{ for all } t \geq 0, \\ \text{and } \dot{x}(\cdot) \text{ is piecewise continuous} \}.$$

We let $ZD(e(\cdot))$ denote the event signal obtained when the (directional) zero detection operation is applied to the signal $e(\cdot)$.

The *forward dynamic mapping* is defined in terms of an initial continuous state x_0 , a discrete state u , a set $T \subset R^{n_x}$, and an event vector $\hat{v} \in V$ as

$$\mathcal{M}_u(x_0 \rightarrow T \uparrow \hat{v}) = \{ x_f \in T \mid \exists x(\cdot) \in \mathcal{X}(x_0, F_u), \text{ and } t_f > 0 \text{ s.t.} \\ x_f = x(t_f), \text{ and for } v(\cdot) = ZD(d - Cx(\cdot)), \\ v(t_f) = \hat{v} \text{ and } v(t) = 0, \forall 0 \leq t < t_f \}.$$

In words, $\mathcal{M}_u(x_0 \rightarrow T \uparrow \hat{v})$, is the set of states in T that are reached by trajectories starting from x_0 , and when reached, the event $\hat{v}$ is generated. Note that the particular event generated when a trajectory reaches T depends on the direction (derivative) of the trajectory at that instant since the zero detector is directional.

We generalize the above definition for a set of initial states $S \in R^{n_x}$ as

$$\mathcal{M}_u(S \rightarrow T \uparrow \hat{v}) = \bigcup_{x_0 \in S} \mathcal{M}_u(x_0 \rightarrow T \uparrow \hat{v}).$$

The *backward dynamic mapping* from T to S under u triggering $\hat{v}$ is defined by

$$\mathcal{B}_u(T \uparrow \hat{v} \rightarrow S) = \{x_0 \in S \mid \mathcal{M}_u(x_0 \rightarrow T \uparrow \hat{v}) \neq \emptyset\}$$

In words, $\mathcal{B}_u(T \uparrow \hat{v} \rightarrow S)$ identifies the part of S from which T can be reached under u , triggering $\hat{v}$ as the first event along the trajectory.

Finally, we define the *null event states* as the set of states from which there exists a trajectory which never triggers any event for a given discrete state u . Formally this set is given as

$$\mathcal{N}_u = \{ x_0 \in R^{n_x} \mid \exists x(\cdot) \in \mathcal{X}(x_0, F_u) \text{ s.t.} \\ v(\cdot) = ZD(d - Cx(\cdot)) \text{ and } v(t) = 0, \forall t \geq 0 \}.$$

States in $\mathcal{N}_u$ can lead to terminating behaviors, that is, to trajectories where no further threshold events are generated by the hybrid system.

4 Approximating Automata

In this section we summarize the method from [2] for constructing an automaton whose behaviors include all possible behaviors of the hybrid system. Let us first discuss the basic ideas of our approach. We start with a finite partition $\mathcal{P}$ of the threshold hyperplanes where each element of $\mathcal{P}$ is connected and completely contained in each element of some subset of the threshold hyperplanes. Given such a partition, we define an approximating automaton A as follows. Except for states labeled q_0 and $q_\perp$, each state of A is labeled by a triple $(P, u, v) \in \mathcal{P} \times U \times V$. There are two types of transitions in the automaton, the *regular event* transitions, each labeled by $(u, v) \in U \times V$, and the *null event* transitions, each labeled by $(u, v_\perp) \in U \times \{v_\perp\}$. A regular event transition (u', v') from (P, u, v) to (P', u', v') is interpreted as follows. Given that a continuous state in

P has been reached under the discrete state u and the event v is triggered at P , there is possibly a continuous trajectory from a state in P to another state in P' under u' that triggers the event v' . The discrete state switching from u to u' must be consistent with the discrete transition function h , that is, $u' \in h(u, v)$. A null event transition $(u', v_\perp)$ from (P, u, v) to $q_\perp$ indicates that there is possibly a continuous trajectory under u' from a state in P that never triggers any event. The *null state* $q_\perp$ is a special state that we use to represent the case where no more threshold events occur. q_0 is the initial state.

To ensure that any subset S of a piece of an event hyperplane P can generate any event that is possible for P , the partition $\mathcal{P}$ needs to be *unambiguous*, which is defined as follows.

Definition 1. Let $M = \bigcup_{i=1, \dots, n_v} M_i$. A partition $\mathcal{P}$ of M is *unambiguous* partition of the threshold hyperplanes if for all $P \in \mathcal{P}$, P is connected and

$$\text{either } P \subseteq M_k \text{ or } P \cap M_k = \emptyset \quad \forall 1 \leq k \leq n_v. \quad (3)$$

In the following we will also say any set $T \subset M$ is unambiguous with respect to the set of hyperplanes $M_1, \dots, M_{n_v}$ if T satisfies the condition (3) with T replacing P .

Definition 2. Given a hybrid system and an unambiguous partition $\mathcal{P}$ of the collection of threshold hyperplanes, we define the *approximating automaton* A corresponding to $\mathcal{P}$ as

$$A = (Q, \Sigma, \delta, q_0)$$

where

Q The set of the states $Q = \mathcal{P} \times U \times V \cup \{q_0, q_\perp\}$

Σ The alphabet is $U \times (V \cup \{v_\perp\})$.

δ The nondeterministic transition function δ for A is defined according to the following rules

1. $(P, u, v) \in \delta(q_0, (u, v))$ if $\mathcal{M}_u(X_0 \rightarrow P \uparrow v) \neq \emptyset$ and $u \in U_0$
2. $q_\perp \in \delta(q_0, (u, v_\perp))$ if $\mathcal{N}_u \cap X_0 \neq \emptyset$ and $u \in U_0$
3. $(P', u', v') \in \delta((P, u, v), (u', v'))$ if $\mathcal{M}_{u'}(P \rightarrow P' \uparrow v') \neq \emptyset$ and $u' \in h(u, v)$
4. $q_\perp \in \delta((P, u, v), (u', v_\perp))$ if $\mathcal{N}_{u'} \cap P \neq \emptyset$ and $u' \in h(u, v)$

q_0 is the initial state of A

Letting L_A denote the collection of all possible finite-length and infinite-length sequences of input-event pairs that can be generated by the automaton A , it can be shown that $L_H \subseteq L_A$ [2]. (In our context, a sequence is *generated* by an automaton A if and only if it is a sequence of labels on either an infinite-length directed path in the automaton starting from q_0 , or the string of event labels on a finite-length path from q_0 to $q_\perp$.) The extent to which A approximates H depends on the granularity of the partition $\mathcal{P}$: if partition $\mathcal{P}'$ refines partition $\mathcal{P}$ with corresponding automata A' and A , then $L_H \subseteq L_{A'} \subseteq L_A$.

We now describe the procedure to obtain a finer partition $\mathcal{P}_{N+1}$ from a given partition $\mathcal{P}_N$. This procedure is shown in Fig. 2. Let A_N be the automaton defined from the current partition $\mathcal{P}_N$. For every hyperplane piece $P \in \mathcal{P}_N$, we identify all the distinct evolutions of the continuous trajectories from P under different u 's. For each transition $(P, u, v) \xrightarrow{(u', v')} (P', u', v')$, we would like to isolate the subset of P corresponding to this transition. We use the backward mapping for this purpose. We can see that the subset we are looking for is exactly the backward mapping $\mathcal{B}_{u'}(P' \uparrow v' \rightarrow P)$. Similarly, for each transition $(P, u, v) \xrightarrow{(u', v_\perp)} q_\perp$, the subset of P corresponding to this transition is simply $\mathcal{N}_{u'} \cap P$.

By computing the backward mapping and the null event set from all successors of states containing P , we have a collection of subsets of P corresponding to different transitions in the automaton A_N . We then make a subpartition of P from these subsets. This process is represented by the routine `disjoint` in Fig. 2. Given a collection of sets, `disjoint` returns a collection of disjoint sets whose union is the same as the union of the input sets. After we have proceeded through all $P \in \mathcal{P}_N$, all subpartitions of the sets P form the new partition $\mathcal{P}_{N+1}$ which is a refinement of $\mathcal{P}_N$. With the new partition, we can define a new automaton A_{N+1} . Therefore, $L_{A_{N+1}} \subseteq L_{A_N}$, i.e., each refinement step gives an approximating automaton that is at least as good an approximation to the hybrid system as the one obtained in the previous step. Furthermore, it can be shown that if $L_{A_{N+1}} = L_{A_N}$, then $L_{A_{N+1}} = L_H$, i.e. if the refinement ever reaches a fixed point then the approximation, in terms of the sequences of input-event pairs, is exact [2].

5 Calculating Mappings and the Null Event Set

The computations required to generate the approximating automata are similar to the computations employed in the tool HyTech [1]. In particular, the fundamental operation is $Reach_F(X)$, which gives the set of continuous states reachable from the set $X \subseteq R^{n_x}$ when the differential inclusion for the continuous-state trajectories is defined by the set $F \subseteq R^{n_x}$. Formally, $Reach_F(X)$ is defined as

$$Reach_F(X) = \bigcup_{x_0 \in X} \{x_f \in R^{n_x} \mid \exists x(\cdot) \in \mathcal{X}(x_0, F) \text{ and } t_f \geq 0 \text{ s.t. } x(t_f) = x_f\}.$$

For a given set $F \subseteq R^{n_x}$, defining

$$-F = \{-s \mid s \in F\},$$

we have that $Reach_{-F}(X)$ corresponds to the *time precondition* operator defined in [1]. $Reach_F(X)$ corresponds to the *time postcondition* operator, which can be defined in a similar manner. Note that $Reach_{-F}(X)$ is the set of states from which the set X can be reached when the differential inclusion is defined by F since $-F$ essentially runs the system backwards in time.

```

 $\mathcal{P}_{N+1} = \text{refine}(\mathcal{P}_N = \{P_1, \dots, P_s\} \mid \mathcal{P}_N \text{ is a partition of } M)$ 
{
   $s = |\mathcal{P}_N|$ 
   $\mathcal{P}' = \emptyset$ 
  for  $i = 1$  to  $s$  {
     $k = 0$ 
    /* check regular event transitions */
    for each transition  $(P_i, u, v) \xrightarrow{(u', v')} (P_j, u', v')$  {
       $S = \mathcal{B}_{u'}(P_j \upharpoonright v' \rightarrow P_i)$ 
      /* add subset of  $P_i$  that maps under  $(u', v')$  to  $P_j$  */
      if  $(S \neq \emptyset)$  {
         $k = k + 1$ 
         $P'_{ik} = S$ 
      }
    }
  }
  /* check null event transitions */
  for each transition  $(P_i, u, v) \xrightarrow{(u', v_\perp)} q_\perp$  {
     $S = \mathcal{N}_u \cap P_i$ 
    /* add subset of  $P_i$  that maps under  $u'$  to  $q_\perp$  */
    if  $(S \neq \emptyset)$  {
       $k = k + 1$ 
       $P'_{ik} = S$ 
    }
  }
  /* call disjoint to compute subpartition of  $P_i$ 
    and form  $\mathcal{P}_{N+1}$  from subpartitions of  $P_i$ 's */
   $\mathcal{P}' = \mathcal{P}' \cup \text{disjoint}(\{P'_{i1}, \dots, P'_{ik}\})$ 
}
return  $\mathcal{P}'$ 
}

```

Fig. 2. Pseudocode for the partition refinement procedure.

The following definitions will also be useful later on in this section. Given an initial state x_0 and a set $F \subseteq R^{n_x}$, the set of states reachable at time t from x_0 under differential inclusion F is defined as

$$Reach_F(t, x_0) = \{x_f \mid \exists x(\cdot) \in \mathcal{X}(x_0, F) \text{ such that } x(t) = x_f\}.$$

For $v \in V$ and T unambiguous with respect to the set of threshold hyperplanes, define the set

$$R_{T \uparrow v} = \bigcap_{i \text{ s.t. } T \subset M_i, v_i=1} M_i^- \bigcap_{i \text{ s.t. } T \subset M_i, \hat{v}_i=0} M_i^{0+}. \quad (4)$$

In words, $R_{T \uparrow v}$ is the set of states from which a state trajectory must arrive at T to generate the event v .

Throughout this section, we assume that a target set T is unambiguous with respect to the set of threshold hyperplanes.

5.1 Forward Mapping

In this subsection, we present the method for computing the forward mapping set under a discrete state u given a set of states S called a *source*, a set of *target* states T , and the target event $\hat{v}$ to be triggered when T is encountered. The forward mapping $\mathcal{M}_u(S \rightarrow T \uparrow \hat{v})$ can be computed as follows.

1. Rule out the part of S for which it is impossible to reach T and trigger $\hat{v}$ before any other event. The remaining part is the candidate initial set.

$$CAN(S \rightarrow T \uparrow \hat{v}) = S \bigcap_{i \text{ s.t. } T \subset M_i^+} M_i^{0+} \bigcap_{i \text{ s.t. } T \subset M_i, \hat{v}_i=0} M_i^{0+}$$

2. Compute the set of via points in $R_{T \uparrow v}$ to which any trajectory from S must pass through in order to reach T and trigger the event $\hat{v}$ before any other events.

$$FVIA_u(S \rightarrow T \uparrow \hat{v}) = Reach_{F_u}(CAN(S \rightarrow T \uparrow \hat{v})) \cap R_{T \uparrow v} \bigcap_{i \text{ s.t. } T \subset M_i^+} M_i^{0+}$$

3. Compute the subset of T that is reachable from $FVIA_u(S \rightarrow T \uparrow \hat{v})$

$$FMAP_u(S \rightarrow T \uparrow \hat{v}) = Reach_{F_u}(FVIA_u(S \rightarrow T \uparrow \hat{v})) \cap T$$

The following lemmas and propositions demonstrate that

$$FMAP_u(S \rightarrow T \uparrow \hat{v}) = \mathcal{M}_u(S \rightarrow T \uparrow \hat{v}).$$

All proofs are given in the Appendix.

Lemma 1. Given two states x_0 and x_f , if there exists an admissible trajectory $x_1(\cdot) \in \mathcal{X}(x_0, F_u)$ reaching x_f at time t_f , there exists another admissible trajectory $x_2(\cdot) \in \mathcal{X}(x_0, F_u)$ which is a straight line connecting x_0 and x_f .

Lemma 2. If a trajectory $x(\cdot) \in \mathcal{X}(x_0, F_u)$ from $x_0 \in S$ to T triggers $\hat{v}$ before any other event, then $x_0 \in CAN(S \rightarrow T \uparrow \hat{v})$.

Proposition 1. Any trajectory $x(\cdot) \in \mathcal{X}(x_0, F_u)$ from a state $x_0 \in S$ that reaches T and triggers $\hat{v}$ before any other event must pass through $FVIA_u(S \rightarrow T \uparrow \hat{v})$.

Proposition 2. If $x_f \in Reach_{F_u}(FVIA_u(S \rightarrow T \uparrow \hat{v})) \cap T$, then for any $x_m \in FVIA_u(S \rightarrow T \uparrow \hat{v})$ such that $x_f \in Reach_{F_u}(\{x_m\})$, there exists a trajectory $x(\cdot) \in \mathcal{X}(x_m, F_u)$ that reaches x_f , triggering $\hat{v}$ before any other event.

Proposition 3. $x_f \in \mathcal{M}_u(S \rightarrow T \uparrow \hat{v}) \iff x_f \in FMAP_u(S \rightarrow T \uparrow \hat{v})$.

5.2 Backward Mapping

The backward mapping under a discrete state u given a source S , target T and, a target event $\hat{v} \in V$. T unambiguous with respect to the threshold hyperplanes.

Given event $\hat{v} \in V$. T unambiguous with respect to the threshold hyperplanes and a discrete state u , we define the backward mapping set $\mathcal{B}_u(T \uparrow \hat{v})$ as

$$\mathcal{B}_u(T \uparrow \hat{v}) = \{ x_0 \mid \exists x(\cdot) \in \mathcal{X}(x_0, F_u), t_f > 0 \text{ s.t. } x(t_f) \in T, \text{ and for } v(\cdot) = ZD(d - Cx(\cdot)), v(t_f) = \hat{v}, \text{ and } v(t) = 0, \forall 0 \leq t < t_f \}$$

The following steps can be used to compute $\mathcal{B}_u(S \rightarrow T \uparrow \hat{v})$.

1. Compute the set of via points in $R_{T \uparrow \hat{v}}$ from which T can be the first subset of threshold hyperplanes that is reached and thereby triggering $\hat{v}$.

$$BVIA_u(T \uparrow \hat{v}) = Reach_{-F_u}(T) \cap R_{T \uparrow \hat{v}} \bigcap_{i \text{ s.t. } T \subset M_i^+} M_i^{0+}$$

2. Compute the set of states from which $BVIA_u(T \uparrow \hat{v})$ can be reached

$$Reach_{-F_u}(BVIA_u(T \uparrow \hat{v}))$$

The above computation gives the set of candidate states from which T can be reached and $\hat{v}$ can be triggered.

3. Restrict the above set by some half spaces to yield the final result

$$BMAP_u(T \uparrow \hat{v}) = Reach_{-F_u}(BVIA_u(T \uparrow \hat{v})) \bigcap_{i, T \subset M_i, \hat{v}_i = 0} M_i^{0+} \bigcap_{i, T \subset M_i^+} M_i^{0+}$$

The propositions that follow lead to our main result in this subsection that $\mathcal{B}_u(T \uparrow \hat{v}) = BMAP_u(T \uparrow \hat{v})$, which is proved in Proposition 6. The backward mapping from $T \uparrow \hat{v}$ back to S under u is simply given by

$$\mathcal{B}_u(T \uparrow \hat{v} \rightarrow S) = \mathcal{B}_u(T \uparrow \hat{v}) \cap S$$

Proposition 4. Any trajectory $x(\cdot) \in \mathcal{X}(x_0, F_u)$ from a state $x_0 \in S$ that reaches T and triggers $\hat{v}$ before any other event must pass through $BVIA_u(T \uparrow \hat{v})$.

Proposition 5. If $x_m \in BVIA_u(T \uparrow \hat{v})$, then there exists a trajectory $x(\cdot) \in \mathcal{X}(x_m, F_u)$ that reaches some $x_f \in T$, triggering $\hat{v}$ before any other event.

Proposition 6. $x_0 \in \mathcal{B}_u(T \uparrow \hat{v}) \iff x_0 \in BMAP_u(T \uparrow \hat{v})$

5.3 Null Event Set

The following proposition gives the fundamental representation of the the set of reachable states for our hybrid systems and for linear hybrid automata. This result provides the basic method for computing the set of null event states.

Proposition 7. Given a continuous state $x_0 \in R^{n_x}$ and a discrete state $u \in U$,

$$Reach_{F_u}(t, x_0) = x_0 + F_u t$$

where

$$x_0 + F_u t = \{x \in R^{n_x} | \exists w \in F_u \text{ s.t. } x = x_0 + wt\}$$

To compute $\mathcal{N}_u$ we define for a state x_0 the set of indices for the threshold hyperplanes for which x_0 lies in the negative halfspace. This set of indices, and the intersection of the associated halfspaces is given as follows.

Definition 3. Given a state x_0 we define I_{x_0} to be the set of indices i of the hyperplanes such that $x_0 \in M_i^-$. We also define S_{x_0} to be the intersection of all such M_i^- . Formally we define,

$$I_{x_0} = \{i | x_0 \in M_i^-\}$$

$$S_{x_0} = \bigcap_{i \in I_{x_0}} M_i^-$$

The significance of I_{x_0} is that threshold events will be *necessarily* generated from a trajectory starting from x_0 only for the threshold hyperplanes with indices in I_{x_0} . The reason is that the directional nature of the zero detector implies that if $i \notin I_{x_0}$, any trajectory crossing M_i from M_i^- must necessarily first cross M_i from M_i^{0+} . However, if such a trajectory exists, there is another trajectory that will cross into M_i and just keep going. That is, it will not return and cross the threshold hyperplane M_i . These observations lead to the following proposition which completely characterizes $\mathcal{N}_u$.

Proposition 8. $x_0 \in \mathcal{N}_u \iff I_{x_0} = \emptyset \text{ or } \exists w \in F_u \text{ s.t. } \forall i \in I_{x_0}, c_i^T w \leq 0$

To calculate $\mathcal{N}_u$, we separate R^{n_x} into 2^{n_v} regions as follows. Let $I \in \{0, 1\}^{n_v}$, and define the set R_I to be

$$R_I = \bigcap_{j:I_j=0} M_j^- \bigcap_{j:I_j=1} M_j^{0+}.$$

For each $I \in \{0, 1\}^{n_v}$, we note that for all $x_0 \in R_I$ the set of indices I_{x_0} is the same. Specifically, $I_{x_0} = \{j | I_j = 0\}$. To decide if $x_0 \in \mathcal{N}_u$ using Proposition 8, we check whether there is a feasible vector w satisfying the following set of constraints

$$\begin{bmatrix} C_{I_{x_0}} \\ A_u \end{bmatrix} w \leq \begin{bmatrix} 0 \\ b_u \end{bmatrix}, \quad (5)$$

where for $I_{x_0} = \{i_1, \dots, i_k\}$, $C_{I_{x_0}}$ is defined as

$$C_{I_{x_0}} = \begin{bmatrix} c_{i_1}^T \\ \vdots \\ c_{i_k}^T \end{bmatrix}$$

If (5) is feasible, then $x_0 \in \mathcal{N}_u$. Since the above check yields the same result for all $x_0 \in R_I$, we have that $R_I \subset \mathcal{N}_u$. After checking all possible I , we have that

$$N_u = \bigcup_{I \text{ s.t. (5) is feasible}} R_I$$

Note that there is no need to check for the case $I = 1_{n_v \times 1}$. In this case $I_{x_0} = \emptyset$ and (5) is always feasible.

6 Example

In this section, we present a simple example to illustrate the construction of approximating automata for a verification problem, and to demonstrate the difference between the approximating automaton approach and the fixed-point reachability computation approach for linear hybrid automata. Consider the *linear hybrid automaton* shown in Figure 3a. We supply this automaton as an input to HyTech and compute the reachability region (for the continuous variables x and y) using the `post` operator, the result for the first three steps are depicted in Figure 3b.

In this example, we would like to verify that starting from the set $S = \{0 \leq x \leq 1, y = 0\}$ in location u_1 , the set $T = \{3 \leq x \leq 4, y = 1\}$ is never reached. It is clear from Figure 3 that T cannot be reached from S . However, by iteratively applying the `post` operator in HyTech, the computation never converges as the reachability region continues to expand in the positive x direction. We now turn to the approximating automata approach.

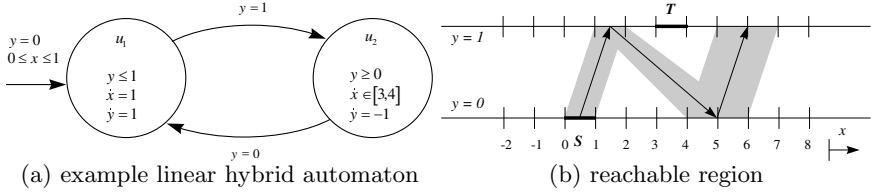


Fig. 3.

The above linear hybrid automaton can be easily converted into our model of hybrid systems introduced in Section 2 by defining the state variables $x_1 = x$ and $x_2 = y$ with the output matrix $C = [0 \ -1; 0 \ 1]$ and the threshold vector $d = [0 \ 1]^T$. The threshold hyperplanes are $-y = 0$ and $y = 1$. The minus sign indicates that the hyperplane $-y = 0$ needs to be hit from above in order for an event to be generated. When the hyperplane $-y = 0$ and $y = 1$ are reached, the events $[1 \ 0]^T$ and $[0 \ 1]^T$ are generated, respectively. The transition function $h(\cdot, \cdot)$ simply switches the differential inclusion to $F_1 = \{\dot{x} = \dot{y} = 1\}$ on the event $[1 \ 0]^T$ and to $F_2 = \{\dot{x} \in [3, 4], \dot{y} = -1\}$ on the event $[0 \ 1]^T$.

To define the first approximating automaton, we select the partition of the threshold hyperplanes shown in Figure 4a. We choose to include S and T in the partition because we would like to analyze their transitional behaviors. The automaton $\mathcal{A}_0$ defined from this partition is also shown in Figure 4a. We drop the discrete state and threshold event from the automaton state labels as it is clear that any subset of $\{-y = 0\}$ is labeled with $u = 2$ and $v = [1 \ 0]^T$ and any subset of $\{y = 1\}$ is labeled with $u = 1$ and $v = [0 \ 1]^T$. The labels on the transitions are also dropped because each of them can be inferred directly from the successor state. From this automaton, we see that T is reachable from S . Thus, we proceed with the automaton refinement procedure.

To simplify our refinement in this example, we will only refine the sets in the partition corresponding to states that are reachable from S . As an example, consider the refinement of B_1 illustrated in Figure 4b. From the transition diagram of $\mathcal{A}_0$, we see that B_1 has multiple successors, namely A_1 , S , and A_2 . We find, through the use of backward mapping, the subsets of B_1 from which A_1 , S , and A_2 can be reached. Using our notation, we compute $\mathcal{B}_2(A_1 \uparrow [1 \ 0]^T \rightarrow B_1)$, $\mathcal{B}_2(S \uparrow [1 \ 0]^T \rightarrow B_1)$, and $\mathcal{B}_2(A_2 \uparrow [1 \ 0]^T \rightarrow B_1)$. These sets are indicated in Figure 4b. After making all of these sets disjoint, we break B_1 into $\{B_{11}, B_{12}, B_{13}, B_{14}\}$ as shown in Figure 4c. We proceed in the similar manner through all the subsets corresponding to the states that are reachable ! from S in automaton $\mathcal{A}_0$ and obtain the next partition and the corresponding automaton $\mathcal{A}_1$ shown in Figure 4c.

In the automaton $\mathcal{A}_1$, T is still reachable from S . We refine the automaton once more and obtain the next partition and the automaton $\mathcal{A}_2$ shown in Figure 4d. In $\mathcal{A}_2$, T is no longer reachable from S and we have our verification result. Therefore, the property that T is not reachable in the hybrid system is veri-

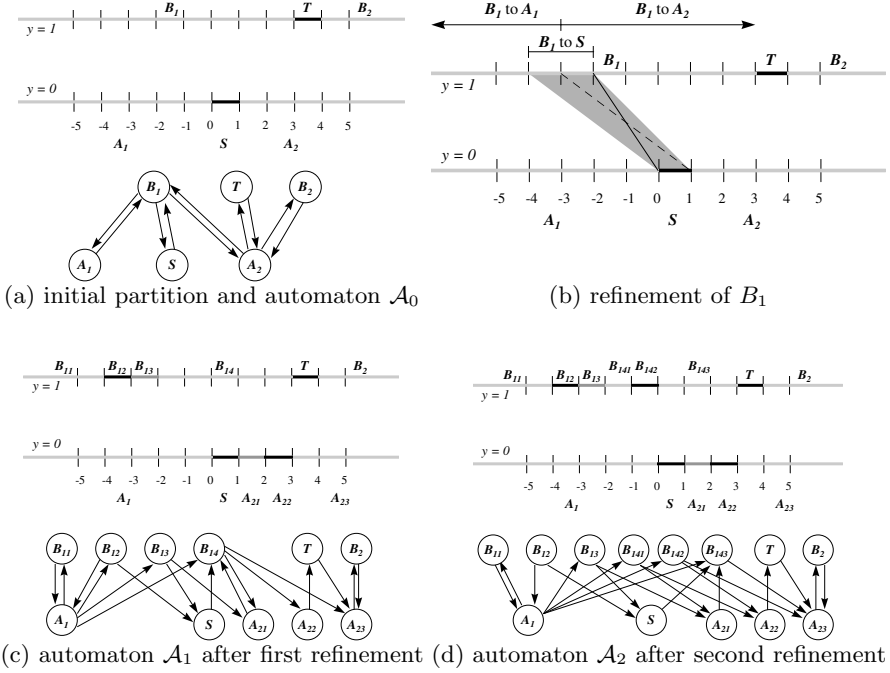


Fig. 4.

fied after two iterations of the approximating automata construction algorithm whereas the HyTech fixed-point computation would never terminate.

7 Discussion

This paper develops a complete set of computations for generating approximating automata for a particular class of hybrid systems characterized by linear constraints on the derivative of the continuous state vector. This demonstrates the possibility of carrying out the steps of the algorithm proposed in [2] (as well as the algorithm in [3]). The example in section 6 illustrates the use of approximating automata for verification and compares the approach to fixed-point computations used in verification tools for hybrid systems.

The hybrid systems considered herein are similar to the linear hybrid automata. One difference between the models is the use of arbitrary discrete dynamics in the hybrid system in Fig. 1. In linear hybrid automata the discrete dynamics are defined a priori as locations which determine the continuous dynamic modes. In the model described in section 2, the separation of the continuous and discrete dynamics allows for a more compact representation of problems in which the particular discrete undetermined. Such situations arise, for example, when the objective is to synthesize a supervisory controller to select the continuous

dynamics to satisfy a specification for the sequential behavior of the system as defined by the threshold events (see [3]).

A more rigorous comparison of the model in section 2 and linear hybrid automata should be made. It is evident an equivalent linear hybrid automaton can always be constructed when the discrete dynamics in Fig. 1 are deterministic. On the other hand, the invariants of linear hybrid automata and nondeterminism in firing times for events defined by guards can be represented by thresholds and freedom in the linear dynamics in the model in Fig. 1. The model in Fig. 1 does not admit jumps in the continuous state, which implies the linear hybrid automata are more general. (A more general model similar to Fig. 1 that admits jumps in the continuous state is described in [5].) In our opinion, the important difference in the models is the perspectives they represent. Linear hybrid automata emphasize the discrete state aspects of the system, whereas the model in Fig. 1 emphasizes the continuous dynamics and a modular, signal-flow view of the world. Each perspective is useful for particular applications.

Current research is focusing on algorithms for constructing approximating automata for hybrid systems with more general continuous dynamics. There are few applications where the continuous dynamics behave according to simple polytope constraints on the derivative of the continuous state vector. We are also investigating the application of approximating automata to the verification and synthesis of switching control laws in safety-critical systems. Results of this research will be reported in future papers.

Appendix: Proofs

Proof of Lemma 1

Suppose $\dot{x}_1(t)$, $0 \leq t \leq t_f$ is piecewise continuous with K pieces. Let $t_0 = 0$ and $0 < t_1 < \dots < t_K$ be the time at the end point of each piece. Note that $t_K = t_f$. For the k^{th} piece, $1 \leq k \leq K$, we apply the Mean Value Theorem to obtain

$$\frac{x_1(t_k) - x_1(t_{k-1})}{t_k - t_{k-1}} = \dot{x}_1(t_k^*), \quad t_{k-1} \leq t_k^* \leq t_k$$

Let $s_k = \dot{x}_1(t_k^*)$ and $x_k = x_1(t_k)$. Note that $x_K = x_1(t_K) = x_1(t_f) = x_f$. We rewrite the above expression as

$$x_k - x_{k-1} = s_k(t_k - t_{k-1})$$

It then follows that

$$\sum_{k=1}^K (x_k - x_{k-1}) = x_K - x_0 = \sum_{k=1}^K s_k(t_k - t_{k-1})$$

Dividing by t_K and letting $\alpha_k = (t_k - t_{k-1})/t_K$, we have that

$$\frac{x_K - x_0}{t_K} = \sum_{k=1}^K \alpha_k s_k$$

Since α_k is nonnegative and

$$\sum_{k=1}^K \alpha_k = \frac{1}{t_K} \sum_{k=1}^K t_k - t_{k-1} = \frac{t_K - t_0}{t_K} = 1,$$

the quantity $s_f = \sum_{k=1}^K \alpha_k s_k$ is a convex combination of s_k . And because $s_k \in F_u \forall 1 \leq k \leq K$, we have that $s_f \in F_u$. Using the above results, we write

$$\frac{x_K - x_0}{t_K} = s_f$$

We construct the trajectory

$$x_2(t) = x_0 + s_f t$$

Recalling that $x_K = x_f$ and $t_K = t_f$, $x_2(t)$ is clearly a straight line connecting x_0 and x_f . It also reaches x_f at t_f . Finally, $x_2(\cdot) \in \mathcal{X}(x_0, F_u)$ because $x_2(0) = x_0$, $\dot{x}_2(t)$ is piecewise continuous (with only one continuous piece), and $\dot{x}_2(t) = s_f \in F_u, \forall t \geq 0$. ■

Proof of Lemma 2 (by contradiction)

Suppose that the trajectory $x(\cdot) \in \mathcal{X}(x_0, F_u)$, where $x_0 \in S$, triggers $\hat{v}$ at $x_f \in T$ before any other event but $x_0 \notin \text{CAN}(S \rightarrow T \uparrow \hat{v})$. Then x_0 must fall into at least one of the following cases.

Case 1 $x_0 \notin S$. This contradicts the assumption that $x_0 \in S$.

Case 2 $x_0 \in \bigcup_i \text{s.t. } T \subset M_i^+ M_i^-$. Consider one threshold hyperplane M_i such that $x_0 \in M_i^-$ and $T \subset M_i^+$. Since T is not contained in M_i , $\hat{v}_i = 0$. Since $x_0 \in M_i^-$ and $T \subset M_i^+$, any trajectory from x_0 to T must cross M_i from the negative side and trigger $\hat{v}_i$ before reaching T , a contradiction.

Case 3 $x_0 \in \bigcup_i \text{s.t. } T \subset M_i, \hat{v}_i = 0$. Consider one threshold hyperplane M_i such that $x_0 \in M_i^-$ and $T \subset M_i, \hat{v}_i = 0$. In order to avoid triggering $\hat{v}_i$, $x(\cdot)$ must approach x_f from M_i^{0+} . However, since $x_0 \in M_i^-$, $x(\cdot)$ has to cross M_i from the negative side before it can enter M_i^{0+} and trigger $\hat{v}_i$, a contradiction. ■

Proof of Proposition 1 (by contradiction)

Suppose there is a trajectory $x(\cdot) \in \mathcal{X}(x_0, u)$ from a state $x_0 \in S$ which reaches T , triggers $\hat{v}$ before any other event but never enters $FVIA_u(S \rightarrow T \uparrow \hat{v})$. Let t_f be time when $\hat{v}$ is triggered, i.e. $x(t_f) \in T$. Then $x(t)$ must remain outside of $FVIA_u(S \rightarrow T \uparrow \hat{v})$ for all $0 \leq t \leq t_f$. It then follows that at any time $0 \leq t_1 < t_f$, at least one of the following is true.

1. $x(t_1) \notin \text{Reach}_{F_u}(\text{CAN}(S \rightarrow T \uparrow \hat{v}))$
2. $x(t_1) \in \bigcup_{i \text{ s.t. } T \subset M_i^+} M_i^-$
3. $x(t_1) \notin R_{T \uparrow v}$

Case 1 $x(t_1) \notin \text{Reach}_{F_u}(\text{CAN}(S \rightarrow T \uparrow \hat{v}))$. This implies that $x_0 \notin \text{CAN}(S \rightarrow T \uparrow \hat{v})$. Then, by Lemma 2, we have that $x(\cdot)$ cannot reach T without triggering some other event before $\hat{v}$, a contradiction.

Case 2 $x(t_1) \in \bigcup_{i \text{ s.t. } T \subset M_i^+} M_i^-$. In this case, there are some indices i such that $T \subset M_i^+$ and $x(t_1) \in M_i^-$. Let $\mathcal{I}$ be the set of all such i . By assumption, $x(t)$ must eventually reach T at time $t_f > t_1$. Since T and $x(t_1)$ bracket M_i for all $i \in \mathcal{I}$, $x(t)$ must have crossed all M_i from the negative side *before* it reaches T . Therefore, $x(t)$ must have generated some other events before reaching T , a contradiction.

Case 3 $x(t_1) \notin R_{T \uparrow v}$. From Case 1 and 2, we have seen that $x(t)$ must remain in $\text{Reach}_{F_u}(S) \cap_{i \text{ s.t. } T \subset M_i^+} M_i^{0+}$ for all $0 \leq t < t_f$ in order to have the possibility to satisfy our assumption. So we assume in this case that $x(t)$ remains in $\text{Reach}_{F_u}(S) \cap_{i \text{ s.t. } T \subset M_i^+} M_i^{0+}$ at all time before reaching T . However, the definition of the event signal $v(\cdot) = ZD(d - Cx(\cdot))$ requires that $x(t)$ enters $R_{T \uparrow \hat{v}}$ just before $\hat{v}$ is triggered. Thus, there exists a time $t_2 > t_1$ such that $x(t_2) \in R_{T \uparrow \hat{v}}$ and $x(t_2) \in \text{Reach}_{F_u}(S) \cap_{i \text{ s.t. } T \subset M_i^+} M_i^{0+}$. Clearly, $x(t_2) \in FVIA_u(S \rightarrow T \uparrow \hat{v})$, a contradiction. ■

Proof of Proposition 2

Suppose x_f and x_m satisfy the above assumption. By Lemma 1, we can construct a straight line trajectory $x(t) = x_m + st$ from x_m to x_f with $x(t_f) = x_f$ for some $t_f > 0$ and $s \in F_u$. We argue that this trajectory never generates any event before it reaches x_f . There are 4 types of event hyperplanes that can be encountered by $x(\cdot)$ during time $0 \leq t < t_f$.

Case 1 M_i s.t. $T \subset M_i$ and $\hat{v}_i = 1$. From the definition of $R_{T \uparrow \hat{v}}$, $x_m \in M_i^-$. Since $x(t)$ is a line going from $x_m \in M_i^-$ to $x_f \in T \subset M_i$, all of these M_i are first reached at t_f from the negative side. Thus, none of these $\hat{v}_i$ are triggered during $0 \leq t < t_f$.

Case 2 M_i s.t. $T \subset M_i$ and $v_i = 0$. From the definition of $R_{T \uparrow \hat{v}}$, $x_m \in M_i^{0+}$. Since $x(t)$ is a line going from $x_m \in M_i^{0+}$ to $x_f \in M_i$, $x(t)$ remains in M_i^{0+} for all $0 \leq t \leq t_f$. Thus, v_i cannot be triggered for all these M_i .

Case 3 M_i s.t. $T \subset M_i^+$. From the definition of $FVIA_u(S \rightarrow T \uparrow \hat{v})$, $x_m \in M_i^{0+}$. Since $x(t)$ is a line going from $x_0 \in M_i^{0+}$ to $x_f \in M_i^+$, $x(t)$ remain in M_i^{0+} for all $0 \leq t \leq t_f$. Thus, $\hat{v}_i$ cannot be triggered for all these M_i .

Case 4 M_i s.t. $T \subset M_i^-$. x_m may be on either side of M_i but $x_f \in M_i^-$. Even if $x(t)$ crosses M_i it does so from the positive side and, therefore, cannot trigger $\hat{v}_i$.

From Case 1, we see that all the events of interest in $\hat{v}$ is triggered for the first time at t_f . From Cases 2, 3, and 4, we see that all the other events are not triggered at all. Thus, $x(t)$ is one such trajectory that reaches T and triggers $\hat{v}$ before any other event. ■

Proof of Proposition 3

$\Rightarrow$ We prove this by contraposition. Suppose $x_f \notin FMAP_u(S \rightarrow T \uparrow \hat{v})$. Then, $x_f \notin Reach_{F_u}(FVIA_u(S \rightarrow T \uparrow \hat{v}))$ or $x_f \notin T$.

Case 1 $x_f \notin T$, then by definition $x_f \notin \mathcal{M}_u(S \rightarrow T \uparrow \hat{v})$.

Case 2 $x_f \notin Reach_{F_u}(FVIA_u(S \rightarrow T \uparrow \hat{v}))$. Assume also for this case that $x_f \in T$. By Proposition 1, any trajectory $x(\cdot) \in \mathcal{X}(x_0, F_u)$ from $x_0 \in S$ which reaches T and triggers $\hat{v}$ before any other event must pass through $FVIA_u(S \rightarrow T \uparrow \hat{v})$. Thus, for x_f to belong to one such trajectory, it must be reachable from $FVIA_u(S \rightarrow T \uparrow \hat{v})$. Thus, there is no admissible trajectory from S that reaches x_f and triggers $\hat{v}$ before any other event and we have that $x_f \notin \mathcal{M}_u(S \rightarrow T \uparrow \hat{v})$.

$\Leftarrow$ Suppose $x_f \in FMAP_u(S \rightarrow T \uparrow \hat{v})$. Since $x_f \in Reach_{F_u}(FVIA_u(S \rightarrow T \uparrow \hat{v}))$ and $FVIA_u(S \rightarrow T \uparrow \hat{v}) \subset Reach_{F_u}(CAN(S \rightarrow T \uparrow \hat{v}))$, there exists a trajectory $x_a(\cdot) \in \mathcal{X}(x_0, u)$ for some $x_0 \in CAN(S \rightarrow T \uparrow \hat{v})$ to $x_f \in T$ via some $x_m \in FVIA_u(S \rightarrow T \uparrow \hat{v})$. Without loss of generality, we assume that x_m is in $\bigcap_i \text{s.t. } T \subset M_i^- M_i^-$ because $x(t)$ must eventually enter $\bigcap_i \text{s.t. } T \subset M_i^- M_i^-$ in order to reach x_f . Since there exists an admissible trajectory from x_0 to x_m , by Lemma 1, we can construct a trajectory $x_1(t) = x_0 + s_m t$ with $x_1(t_m) = x_0 + s_m t_m = x_m$, which is a straight line connecting x_0 and x_m . There are 4 types of threshold hyperplanes that might be encountered by $x_1(t)$.

Case 1 M_i such that $T \subset M_i^+$. For each of these hyperplanes, both x_0 and x_m are in M_i^{0+} by the definition of $CAN(S \rightarrow T \uparrow \hat{v})$ and $FVIA_u(S \rightarrow T \uparrow \hat{v})$. Thus, the whole line connecting them is also contained in M_i^{0+} for each M_i . Consequently, $x_1(t)$ cannot trigger the event $\hat{v}_i$ corresponding to any of these M_i for all $0 \leq t \leq t_m$.

Case 2 M_i such that $T \subset M_i^-$. As discussed above, $x_m \in M_i^-$ for each of these M_i . Therefore, even if $x_0 \in M_i^+$ for some M_i , $x_1(t)$ crosses all of such M_i from the positive side. Thus, $x_1(t)$ cannot trigger the event $\hat{v}_i$ corresponding to any of these M_i for all $0 \leq t \leq t_m$.

Case 3 M_i such that $T \subset M_i, \hat{v}_i = 0$. From the definition of $CAN(S \rightarrow T \uparrow \hat{v})$ and $FVIA_u(S \rightarrow T \uparrow \hat{v})$, for each of these M_i , both x_0 and x_m are in M_i^{0+} and so is the whole line connecting them. Thus, $x_1(t)$ cannot trigger the event $\hat{v}_i$ corresponding to any of these M_i for all $0 \leq t \leq t_m$.

Case 4 M_i such that $T \subset M_i, \hat{v}_i = 1$. From the definition of $FVIA_u(S \rightarrow T \uparrow \hat{v})$, $x_m \in M_i^-$ for all these hyperplanes. By the same argument as the one in Case 2, we have that $x_1(t)$ cannot trigger the event $\hat{v}_i$ corresponding to any of these M_i for all $0 \leq t \leq t_m$.

From Case 1 to 4, we have that the trajectory $x_1(t)$ cannot trigger any event up to time t_m .

Since $x_m \in FVIA_u(S \rightarrow T \uparrow \hat{v})$ and $x_f \in Reach_{F_u}(\{x_m\})$, there exists a trajectory $x_2(t) \in \mathcal{X}(x_m, u)$ which reaches $x_f \in T$ and triggers $\hat{v}$ before any other event by Proposition 2. Thus, we can construct a single trajectory that

triggers $\hat{v}$ before any other event from x_0 to x_f via x_m by combining $x_1(t)$ and $x_2(t)$ as shown below.

$$x(t) = \begin{cases} x_1(t) & 0 \leq t < t_m \\ x_2(t - t_m, x_m) & t \geq t_m \end{cases}$$

The above result implies the left hand side of Proposition 3. ■

Proof of Proposition 4 (by contradiction)

Suppose there is a trajectory $x(\cdot) \in \mathcal{X}(x_0, F_u)$ which reaches T , triggers $\hat{v}$ before any other event, but never enters $BVIA_u(T \uparrow \hat{v})$. Let t_f be the time when $\hat{v}$ is triggered. Then $x(t)$ must remain outside of $BVIA_u(T \uparrow \hat{v})$ for all $0 \leq t < t_f$. In other words, at any time $0 \leq t_1 < t_f$, at least one of the following is true.

1. $x(t_1) \notin Reach_{-F_u}(T)$
2. $x(t_1) \in \bigcup_{i, T \subset M_i^+} M_i^-$
3. $x(t_1) \notin R_{T \uparrow \hat{v}}$

Case 1 $x(t_1) \notin Reach_{-F_u}(T)$. In this case $x(t)$ cannot reach T for all $t \geq t_1$, a contradiction.

Case 2 $x(t_1) \in \bigcup_{i, T \subset M_i^+} M_i^-$. In this case, we have a contradiction by the same argument as discussed in Case 2 of the proof of Proposition 1.

Case 3 $x(t_1) \notin R_{T \uparrow \hat{v}}$. In this case, we have a contradiction by the same argument as discussed in Case 3 of the proof of Proposition 1. ■

Proof of Proposition 5

Suppose $x_m \in BVIA_u(T \uparrow \hat{v})$. Since x_m must be in $Reach_{-F_u}(T)$, Lemma 1 implies that there exists a state $x_f \in T$, a vector $-s \in -F_u$, and a trajectory $x_r(t) = x_f - st$ such that $x_r(t_f) = x_m$, which is a straight line connecting x_f and x_m . Since $-s \in -F_u \Rightarrow s \in F_u$, we can construct a forward straight line trajectory $x(t) = x_m + st$ from x_m to x_f with $x(t_f) = x_f$. This trajectory never generates any event before it reaches x_f by the same argument discussed in the proof of Proposition 2. ■

Proof of Proposition 6

$\Rightarrow$ We prove this by contraposition. Suppose $x_0 \notin BMAP_u(T \uparrow \hat{v})$, then x_0 must fall into at least one of the following cases.

Case 1 $x_0 \notin Reach_{-F_u}(BVIA_u(T \uparrow \hat{v}))$. In this case, any trajectory from x_0 can never reach a state in $BVIA_u(T \uparrow \hat{v})$. By Proposition 4, such trajectory cannot reach T and trigger $\hat{v}$ before any other event.

Case 2 $x_0 \in \bigcup_{i \text{ s.t. } T \subset M_i^+} M_i^-$. In this case, no trajectory from x_0 can reach T and trigger $\hat{v}$ before any other event (see Case 2 in the proof of Lemma 2).

Case 3 $x_0 \in \bigcup_i \text{s.t. } T \subset M_i, \hat{v}_i=0 \ M_i^-$. In this case, no trajectory from x_0 can reach T and trigger $\hat{v}$ before any other event (see Case 3 in the proof of Lemma 2).

$\Leftarrow$ Suppose $x_0 \in BMAP_u(T \uparrow \hat{v})$.

If x_0 is also in $VIA_u(T \uparrow \hat{v})$, the implication follows from Proposition 5.

Now suppose $x_0 \notin VIA_u(T \uparrow \hat{v})$. Since $x_0 \in Reach_{-F_u}(VIA_u(T \uparrow \hat{v}))$ and $VIA_u(T \uparrow \hat{v}) \subset Reach_{-F_u}(T)$, there exists a trajectory $x_a(\cdot) \in \mathcal{X}(x_0, F_u)$ from x_0 to $x_f \in T$ via $x_m \in VIA_u(T \uparrow \hat{v})$. Without loss of generality, we assume that x_m is in $\bigcap_{i, T \subset M_i^-} M_i^-$. This is because $x_a(\cdot)$ must eventually enter $\bigcap_{i, T \subset M_i^-} M_i^-$ in order to reach T . We can construct a two-piece linear trajectory from x_0 to x_m and from x_m to x_f and show that this trajectory triggers $\hat{v}$ before any other event at x_f as discussed in the Proof of Proposition 3. ■

Proof of Proposition 7

We prove that $x \in Reach_{F_u}(t_1, x_0) \iff x \in x_0 + F_u t_1$

$\Rightarrow$ Suppose $x \in Reach_{F_u}(t_1, x_0)$. Then there exists a trajectory $x_1(\cdot) \in \mathcal{X}(x_0, u)$ such that $x = x_1(t_1)$. By Lemma 1, there exists another admissible trajectory $x_2(t) = x_0 + wt$ such that $x = x_0 + wt_1$ for some $w \in F_u$. Thus, we have that $x \in x_0 + F_u t_1$.

$\Leftarrow$ Suppose $x \in x_0 + F_u t_1$. Then $x = x_0 + wt_1$ for some $w \in F_u$. We construct a trajectory $x(t) = x_0 + wt$. Clearly, $x(\cdot) \in \mathcal{X}(x_0, u)$ and $x(t_1) = x$, so we have that $x \in Reach_{F_u}(t_1, x_0)$. ■

Proof of Proposition 8

$\Rightarrow$ We prove this implication by contradiction. Suppose that $I_{x_0} \neq \emptyset$ and $\forall w \in F_u, \exists i \in I_{x_0}$ s.t. $c_i^T w > 0$. For each w , we pick one such index and denote it by i_w . Since $c_{i_w}^T x_0 < d_{i_w}$ (by Definition 3) and $c_{i_w}^T w > 0$, there exists a time t_w such that for $t \geq t_w$, $c_{i_w}^T (x_0 + wt) \geq d_{i_w}$ or, equivalently, $x_0 + wt \notin M_{i_w}^-$. Specifically, define

$$t_w = \frac{d_{i_w} - c_{i_w}^T x_0}{c_{i_w}^T w}$$

Now, let

$$t^* = \max_{w \in F_u} t_w$$

It is clear that for all $w \in F_u$, $c_{i_w}^T (x_0 + wt^*) \geq d_{i_w}$. It follows that for all $w \in F_u$, there exists an index $i \in I_{x_0}$, namely i_w , such that $x_0 + wt^* \notin M_i^-$. This, in turns, implies that for all $w \in F_u$, $x_0 + wt^* \notin S_{x_0}$. By Proposition 7, we have that $x_0 + F_u t^* = Reach_{F_u}(t^*, x_0) \subset \overline{S_{x_0}}$. Thus, for any trajectory $x(\cdot) \in \mathcal{X}(x_0, u)$, $x(t^*) \in M_i^{0+}$ for some $i \in I_{x_0}$. But $x(0) = x_0 \in M_i^-$ by Definition 3, so $x(\cdot)$ must have crossed M_i from the negative side, triggering the event v_i , before or at t^* . Therefore, we conclude that $x_0 \notin \mathcal{N}_u$.

←

Case 1 If $I_{x_0} = \emptyset$, then $x_0 \in M_i^{0+}$ for all i . We choose any vector $w \in F_u$ and construct an admissible trajectory $x(t) = x_0 + wt$. This trajectory can never generate any event. This is because even if $x(\cdot)$ ever crosses any hyperplane M_i , it does so from the M_i^+ side and, therefore, cannot generate the event v_i .

Case 2 If $\exists w \in F_u$ s.t. $\forall i \in I_{x_0}, c_i^T w \leq 0$, we choose one such w and construct an admissible trajectory $x(t) = x_0 + wt$. We argue that this trajectory can never generate any event. For $i \notin I_{x_0}$, $x_0 \in M_i^{0+}$. The trajectory can never generate the event v_i by the same reasoning as in Case 1. For $i \in I_{x_0}$, $c_i^T x_0 < d_i$. Since $c_i^T \dot{x}(t) = c_i^T w \leq 0$, $c_i^T x(t) < d_i$ for all $t \geq 0$. Consequently, $x(t)$ can never generate the event v_i . Thus, for all i , $x(\cdot)$ can never generate v_i .

From Case 1 and Case 2, we conclude that $x_0 \in \mathcal{N}_u$. ■

References

1. R. Alur, T.A. Henzinger, and P.-H. Ho. Automatic symbolic verification of embedded systems. *IEEE Trans. on Software Engineering*, 22(3):181–201, Mar 1996.
2. A. Chutinan and B.H. Krogh. Computing approximating automata for a class of hybrid systems. *Mathematical Modeling of Systems: Special Issue on Discrete Event Models of Continuous Systems*, submitted.
3. J.E.R. Cury, B.H. Krogh, and T. Niinomi. Synthesis of supervisory controllers for hybrid systems based on approximating automata. *IEEE Trans. on Automatic Control*, to appear in the Special Issue on Hybrid Systems, 1997. a preliminary version of this paper also appeared in *Proc. 34th IEEE CDC*, pp. 1461–1466.
4. P.W. Glynn. A gsmf formalism for discrete event systems. *Proceedings of the IEEE*, 77(1):14–23, Jan 1989.
5. B.H. Krogh. Condition/event signal interfaces for block diagram modeling and analysis of hybrid systems. In *Proc. 1993 IEEE International Symposium on Intelligent Control*, pages 180–185, Aug 1993.
6. R.P. Kurshan. *Computer-Aided Verification of Coordinating Processes*. Princeton University Press, Princeton, NJ, 1994.
7. K.L. McMillan. *Neuro-Dynamic Programming*. Klumer, 1993.
8. T. Niinomi and B.K. Krogh. Modeling and analysis of switched-mode hybrid systems driven by threshold events. In H. Khalil, J. Chow, and P.A. Ioannou, editors, *Proc. of Workshop on Advances in Control Systems and its Applications*, pages 155–172. Springer-Verlag, 1995.
9. T. Niinomi, B.K. Krogh, and J.E.R. Cury. Refinements of approximating automata for synthesis of supervisory controllers for hybrid systems. In R. Alur, T.A. Henzinger, and E.D. Sontag, editors, *Hybrid Systems III*, pages 475–484. Springer-Verlag, 1996.
10. J. Raisch and S.D. O’Young. Discrete approximation and supervisory control of continuous systems. *IEEE Trans. on Automatic Control*, page to appear, 1998.
11. P.J.G. Ramadge and W.M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81–98, Jan 1989.

On Hybrid Systems and the Modal μ -calculus

J. M. Davoren*

Center for Foundations of Intelligent Systems
626 Rhodes Hall, Cornell University
Ithaca NY 14853, USA
davoren@hybrid.cornell.edu

Abstract. We start from a basic and fruitful idea in current work on the formal analysis and verification of hybrid and real-time systems: the *uniform* representation of both sorts of state dynamics – both continuous evolution within a control mode, and the effect of discrete jumps between control modes – as *abstract transition relations* over a hybrid space $X \subseteq Q \times \mathbb{R}^n$, where Q is a finite set of control modes. The resulting “machine” or *transition system model* is currently analyzed using the resources of concurrent and reactive systems theory and temporal logic verification, abstracted from their original setting of finite state spaces and purely discrete transitions. One such resource is the *propositional μ -calculus*: a richly expressive formal logic of transition system models (of arbitrary cardinality), which subsumes virtually all temporal and modal logics. The key move here is to view the transition system models of hybrid automata not merely as some form of “discrete abstraction”, but rather as a skeleton which can be fleshed out by imbuing the state space with *topological*, *metric tolerance* or other *structure*. Drawing on the resources of modal logics, we give explicit symbolic representation to such structure in polymodal logics extending the modal μ -calculus. The result is a logical formalism in which we can directly and simply express *continuity properties of transition relations* and metric tolerance properties such as “being within distance ϵ ” of a set. Moreover, the logics have sound and complete deductive proof systems, so assumptions of continuity or tolerance can be used as hypotheses in deductive verification. By also viewing transition relations in their equivalent form as *set-valued functions*, and drawing on the resources of set-valued analysis and dynamical systems theory, we open the way to a richer formal analysis of robustness and stability for hybrid automata and related classes of systems.

1 Introduction

It is hardly controversial to claim that the μ -calculus is a formal logic of central import for the analysis and verification of hybrid automata and related classes of systems. The fundamental concepts of *reachability* and *invariance* are expressible

* Research supported by the ARO under the MURI program “Integrated Approach to Intelligent Systems”, grant no. DAA H04-96-1-0341.

in terms of fixed-points of operators mapping sets of states to sets of states, and are thus definable in the language of the μ -calculus. The iterative computation of the denotation of such fixed point formulas lies at the heart of symbolic model checking tools for hybrid and real-time systems such as HYTECH [4], [19] and KRONOS [13]. More generally, the propositional μ -calculus is well-recognized as a richly expressive logic over transition system models: the power of its fixed-point quantifiers are such that it subsumes virtually all temporal, modal and dynamic logics [15], [25].

However, the current practice, within the larger field of automated verification of (discrete) reactive systems as well as within the hybrid systems community, is to consider the μ -calculus not as a working or usable logic but rather as a logic of the substratum. It provides a common “machine” language and semantics for verification by model checking, with user-input specifications written in the more “natural” languages of temporal logics, and then translated into that of the μ -calculus.

This paper challenges that practice, and demonstrates that the propositional μ -calculus and various of its modal logic extensions can provide both an expressively rich and “human readable” formalism for reasoning about properties of hybrid dynamical systems.

We begin with the “machine” or transition system models of hybrid systems, in which both sorts of state transformation – continuous evolution within a control mode, and the effects of discrete jumps between control modes – are uniformly represented as *abstract transition relations* $r \subseteq X \times X$ over a hybrid state space $X \subseteq Q \times \mathbb{R}^n$, where Q is a finite set of control modes or discrete states.

Formally, define a *labeled transition system* (LTS) (or *generalized Kripke model*) to be a structure

$$\mathfrak{M} = \left(X, \{a^{\mathfrak{M}}\}_{a \in \Sigma}, \{\|p\|^{\mathfrak{M}}\}_{p \in \Phi} \right) \quad (1)$$

where $X \neq \emptyset$ is the state space (of arbitrary cardinality); for each transition label $a \in \Sigma$, $a^{\mathfrak{M}} \subseteq X \times X$ is a binary relation on X ; and for each propositional constant (observation or event label) $p \in \Phi$, $\|p\|^{\mathfrak{M}} \subseteq X$ is a fixed subset of X .

An LTS model is a clean and simple abstraction of a *finite automaton*. Such an $\mathfrak{M}$ is an abstract machine over state space X , with input or action alphabet Σ and transition map $\delta : X \times \Sigma \rightarrow \mathcal{P}(X)$ given by: $x' \in \delta(x, a)$ iff $x \xrightarrow{a^{\mathfrak{M}}} x'$. It is additionally equipped with an observation alphabet Φ , and an output map $o : X \rightarrow \mathcal{P}(\Phi)$ given by: $o(x) = \{p \in \Phi \mid x \in \|p\|^{\mathfrak{M}}\}$; sets of initial or final states can be identified by specific labels in Φ .

A (basic) hybrid automata $\mathcal{H}$ is typically represented by a graph of the form depicted in Figure 1. Hybrid automata and their associated LTS models are examined in more detail in Section 2; for now, we give a high-level description, based on Henzinger’s “time-abstract” transition system in [19] §1.2.

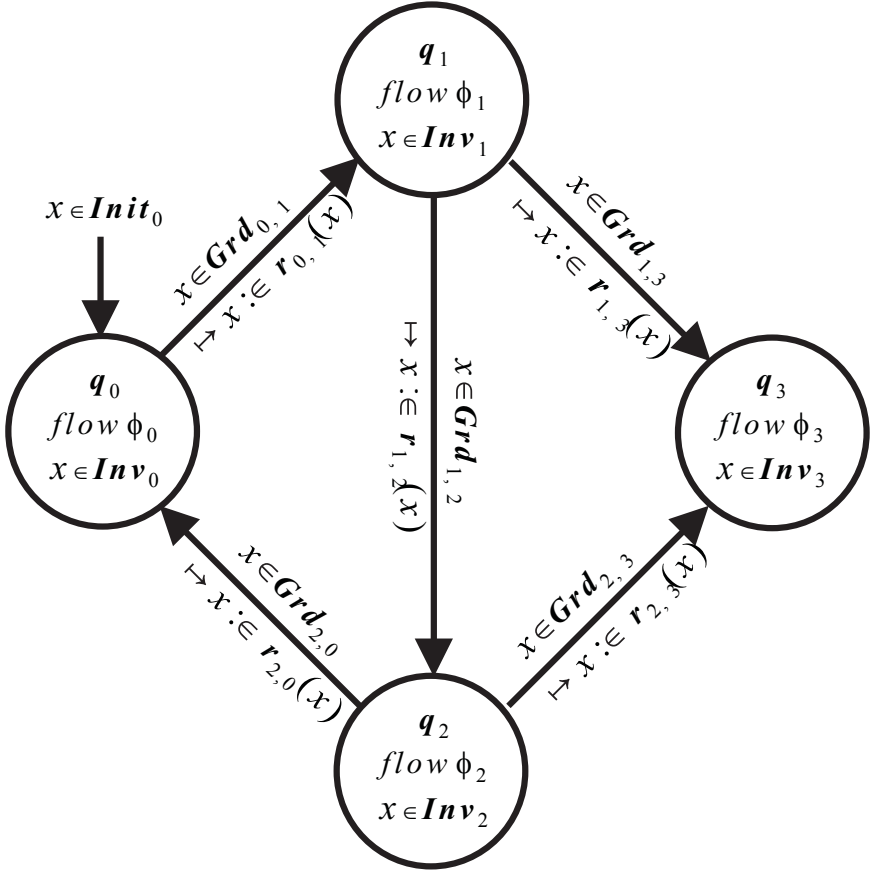


Fig. 1. Basic hybrid automaton

An LTS model $\mathfrak{M}_{\mathcal{H}}$ of a hybrid automaton $\mathcal{H}$ has a state space $X \subseteq Q \times \mathbb{R}^n$, with Q finite. So states are pairs (q, x) , where $q \in Q$ and $x = (x_1, \dots, x_n) \in \mathbb{R}^n$. For each $q \in Q$, let $X_q \subseteq \mathbb{R}^n$ be the projection of X under q . The transition alphabet Σ will include symbols such as e_q for the relation of *evolution* (a “time-step” or “continuous transition”) within each discrete mode $q \in Q$. In the basic case, such a relation is defined by: $(q, x) \xrightarrow{e_q} (q, x')$ iff there is an integral curve along the flow ϕ_q connecting $x \in X_q$ to $x' \in X_q$, and all points on the curve between x and x' lie inside the invariant set $Inv_q \subseteq X_q$. The transition alphabet will also include, for each edge (q, q') in the discrete transition graph $G \subseteq Q \times Q$ of $\mathcal{H}$, a symbol $c_{q,q'}$ for the *controlled jump* relation (a “step” or “discrete transition”) modeling the effect of making a controlled switch from mode q to mode q' . Such relations are standardly defined by: $(q, x) \xrightarrow{c_{q,q'}} (q', x')$ iff $x \in Grd_{q,q'}$, $x' \in Inv_{q'}$, and $x' \in r_{q,q'}(x)$, where $r_{q,q'} \subseteq X_q \times X_{q'}$ is a reset relation for the real-valued coordinates, and the domain $Grd_{q,q'} \subseteq X_q$ is known

as the guard set of the discrete transition (q, q') . The alphabet Φ of atomic propositions will include **Init** $_q$ and **Inv** $_q$ for $q \in Q$, and **Grd** $_{q,q'}$ for $(q, q') \in G$.

A trajectory of $\mathcal{H}$ is a finite or infinite sequence $\langle \delta_i, q_i, \gamma_i \rangle_{i \in I}$ such that for each $i \in I$: the duration $\delta_i \geq 0$; the curve $\gamma_i : [0, \delta_i] \rightarrow X_{q_i}$ is such that $(q_i, \gamma_i(0)) \xrightarrow{e_{q_i}} (q_i, \gamma_i(t))$ for all $t \in [0, \delta_i]$; $(q_i, q_{i+1}) \in G$; and $(q_i, \gamma_i(\delta_i)) \xrightarrow{c_{q_i, q_{i+1}}} (q_{i+1}, \gamma_{i+1}(0))$. When I is finite, with largest element N , it is allowed that $\delta_N = \infty$. When a hybrid automaton is thought of as a discrete controller interacting with a physical plant, the class of trajectories, so defined, are founded on implicit operational assumptions of continuous and perfect precision sensing, and instantaneous control switches ([19]).

In the *modal* – as distinct from *temporal* – variant of the μ -calculus¹, the propositional language (over an alphabet (Σ, Φ)) includes a dual pair of modal operators $[a]$ and $\langle a \rangle$, for each transition label $a \in \Sigma$. The (standard) relational Kripke semantics of the labeled modalities are given by the *universal* and *existential pre-image operators* of the corresponding relations $r = a^{\mathfrak{M}}$. For relations $r \subseteq X \times Y$, and sets $A \subseteq Y$,

$$\begin{aligned} \tau(r)(A) &\doteq \{ x \in X \mid (\forall y \in Y)[x \xrightarrow{r} y \Rightarrow y \in A] \} \\ \sigma(r)(A) &\doteq \{ x \in X \mid (\exists y \in Y)[x \xrightarrow{r} y \wedge y \in A] \} \end{aligned} \quad (2)$$

In the notation of [20], $\sigma(r) = \text{pre}[r]$ and $\tau(r) = \widetilde{\text{pre}}[r]$. The semantic readings of the modalities are *forward*-looking, and in temporal logics, they are known as relativized *next* operators:

$$\begin{aligned} [a]\varphi &\Rightarrow \text{“All } a\text{-successors satisfy } \varphi\text{”} \\ \langle a \rangle \varphi &\Rightarrow \text{“Some } a\text{-successor satisfies } \varphi\text{”} \end{aligned}$$

The temporal variant of the μ -calculus usually works with the *global* transition relation $R^{\mathfrak{M}} = \bigcup_{a \in \Sigma} a^{\mathfrak{M}}$ (standardly assumed to be *total*) and the modal operators are replaced by global temporal “next” operators: $\forall X$ or $\forall \bigcirc$, and $\exists X$ or $\exists \bigcirc$.

Sentences φ of the μ -calculus denote *sets of states* $\|\varphi\|^{\mathfrak{M}} \subseteq X$, and a sentence is *true* in $\mathfrak{M}$, written $\mathfrak{M} \models \varphi$, iff $\|\varphi\|^{\mathfrak{M}} = X$, or equivalently, $\|\neg\varphi\|^{\mathfrak{M}} = \emptyset$. The propositional connectives $\neg$, $\wedge$ and $\vee$ are interpreted by set theoretic complement, intersection and union, and other connectives and constants defined in the usual way. In particular, $\|\mathbf{tt}\|^{\mathfrak{M}} = X$, and an implication $\varphi \rightarrow \psi$ is true in $\mathfrak{M}$ exactly when $\|\varphi\|^{\mathfrak{M}} \subseteq \|\psi\|^{\mathfrak{M}}$. As a point of contrast, in the language of *linear temporal logic* **LTL**, sentences denote sets of (finite or infinite) *paths* or *trajectories* of the LTS model, rather than sets of states. In the language of the branching temporal logic **CTL**^{*}, there are two sorts of sentences: state sentences, true or false at states of the LTS model, and path sentences, true or

¹ The formal syntax and semantics of the μ -calculus are reviewed in detail in Section 3 below. For an account of the modal and temporal flavors of the μ -calculus, see [38] §4.2. [15] is a good source for translations of various linear and branching time temporal logics into the μ -calculus. For background on modal logics, see [9], [35].

false of infinite paths through the model. An $\exists$ or $\forall$ path quantifier applied to a path sentence produces a state sentence, and such quantification is definable using the least and greatest fixed-point quantifiers of the μ -calculus.

The principal advantage of working in the modal rather than temporal framework is that it gives a *modular* specification language for expressing properties of transition systems: we can describe and reason about each of the *component* transition relations of an LTS model, and how they are combined to form more complex transition relations. In particular, we can give a clean and modular formal description of classes of trajectories of the system.

The modal sentences:

$$\psi \rightarrow [\mathbf{c}_{q,q'}]\varphi \quad \text{and} \quad \psi \rightarrow [\mathbf{e}_q]\varphi$$

with the semantic readings “If ψ holds, then all $\mathbf{c}_{q,q'}$ -successors satisfy φ ”, and likewise for $\mathbf{e}_q$, correspond precisely to Manna and Pnueli’s two types of (temporal logic) safety verification conditions for hybrid systems in [29] §4.1. Their notation is: $\{\psi\}\tau\{\varphi\}$ and $\{\psi\}cont\{\varphi\}$, respectively, where τ ranges over jump transitions and “*cont*” denotes the union of all the evolution relations.

The modal sentence

$$\langle \mathbf{e}_{q_0} \rangle \langle \mathbf{c}_{q_0,q_1} \rangle \langle \mathbf{e}_{q_1} \rangle \langle \mathbf{c}_{q_1,q_2} \rangle \langle \mathbf{e}_{q_2} \rangle \cdots \langle \mathbf{e}_{q_{k-1}} \rangle \langle \mathbf{c}_{q_{k-1},q_k} \rangle \langle \mathbf{e}_{q_k} \rangle \varphi \quad (3)$$

denotes the set of states (q_0, x) from which *some* trajectory with discrete trace $(q_0, q_1, \dots, q_k)$ *reaches* the set $\|\varphi\|^{\mathfrak{M}} \subseteq X$. Dually, the modal sentence

$$[\mathbf{e}_{q_0}][\mathbf{c}_{q_0,q_1}][\mathbf{e}_{q_1}][\mathbf{c}_{q_1,q_2}][\mathbf{e}_{q_2}] \cdots [\mathbf{e}_{q_{k-1}}][\mathbf{c}_{q_{k-1},q_k}][\mathbf{e}_{q_k}] \varphi \quad (4)$$

denotes the set of states from which *all* $(q_0, q_1, \dots, q_k)$ -trajectories *reach* the set $\|\varphi\|^{\mathfrak{M}}$, upon the last jump c_{q_{k-1},q_k} and remain in $\|\varphi\|^{\mathfrak{M}}$ throughout the last evolution e_{q_k} .

Defining $\mathbf{e}$ and $\mathbf{c}$ to denote the relational sum (union) of, respectively, the relations for the $\mathbf{e}_q$ ’s for $q \in Q$, and the relations for the $\mathbf{c}_{q,q'}$ ’s for $(q, q') \in G$, the dynamics of the class of all hybrid trajectories with finite discrete traces are captured by the dual fixed-point definable modalities:

$$\langle \mathbf{h} \rangle \varphi \doteq \mu Z. \langle \mathbf{e} \rangle \varphi \vee \langle \mathbf{e} \rangle \langle \mathbf{c} \rangle Z \quad \text{and} \quad [\mathbf{h}] \varphi \doteq \nu Z. [\mathbf{e}] \varphi \wedge [\mathbf{e}] \langle \mathbf{c} \rangle Z \quad (5)$$

The sentence $\langle \mathbf{h} \rangle \varphi$ “unwinds” to the infinite union of all sentences of the form (3), and dually, $[\mathbf{h}] \varphi$ corresponds to the intersection of all sentences of the form (4). As a regular expression, we have $\mathbf{h} = (\mathbf{ec})^* \mathbf{e} = \mathbf{e}(\mathbf{ce})^*$ (so we are in fact working in the weaker propositional dynamic logic **PDL**, rather than the full μ -calculus.) Semantically, $\langle \mathbf{h} \rangle$ and $[\mathbf{h}]$ correspond to the dual pre-image operators of the *reachability relation* h of the system under the control of $\mathcal{H}$; that is, $(q, x) \xrightarrow{h} (q', x')$ iff some trajectory $\langle \delta_i, q_i, \gamma_i \rangle_{i \in I}$ with $q_0 = q$ and $\gamma_0(0) = x$ passes through the point (q', x') .

We now have the formal linguistic machinery to succinctly express various system specifications. The *safety* sentence

$$\mathbf{Init} \rightarrow [\mathbf{h}] \varphi \quad (6)$$

is *true* in the model $\mathfrak{M} = \mathfrak{M}_{\mathcal{H}}$ exactly when every trajectory that starts in the set $\|\mathbf{Init}\|^{\mathfrak{M}}$ *always* remains within $\|\varphi\|^{\mathfrak{M}}$. More generally, we say a set $\|\varphi\|^{\mathfrak{M}}$ is *future-invariant* under $\mathcal{H}$ exactly when the sentence $\varphi \rightarrow [\mathbf{h}]\varphi$ is true in $\mathfrak{M}$. We also have at our disposal (previously unutilized) *deductive proof systems* for the μ -calculus, such as Kozen’s axiomatization $\mathbf{L}_{\mu}$ [23], [5], [40], which is sound and complete over *arbitrary LTS* models. From the fixed-point rules of $\mathbf{L}_{\mu}$ (given in Section 5), one readily derives an obvious *invariance rule* for hybrid trajectories:

$$\frac{\psi \rightarrow \varphi \quad \varphi \rightarrow [\mathbf{e}_q]\varphi \quad \varphi \rightarrow [\mathbf{c}_{q,q'}]\varphi}{\psi \rightarrow [\mathbf{h}]\varphi} \quad \text{for } q \in Q, (q, q') \in G \quad (7)$$

This is a simpler μ -calculus analog of the **LTL** invariance rule used in the verification of safety properties for hybrid automata in [29], [30].

To express liveness properties, we use modal analogs of the “box-diamond” construct in temporal logic. For example, the sentence

$$\varphi \rightarrow [\mathbf{h}]\langle \mathbf{e} \rangle \langle \mathbf{c} \rangle \langle \mathbf{e} \rangle \mathbf{tt} \quad (8)$$

is true in $\mathfrak{M}$ exactly when every maximal $\mathcal{H}$ trajectory from a state in $\|\varphi\|^{\mathfrak{M}}$ has an *infinite* discrete trace. This is so because $[\mathbf{h}]\langle \mathbf{e} \rangle \langle \mathbf{c} \rangle \langle \mathbf{e} \rangle \mathbf{tt}$ denotes the set of states from which every trajectory with a finite discrete trace can be properly extended. Similarly, the sentence $\varphi \rightarrow [\mathbf{h}]\langle \mathbf{e} \rangle \langle \mathbf{c} \rangle \langle \mathbf{e} \rangle \varphi$ is true in $\mathfrak{M}$ exactly when every trajectory from $\|\varphi\|^{\mathfrak{M}}$ returns to $\|\varphi\|^{\mathfrak{M}}$ via a controlled jump *infinitely often*. And $[\mathbf{h}]\langle \mathbf{h} \rangle \varphi$ denotes the set of states from which every hybrid trajectory *eventually* reaches $\|\varphi\|^{\mathfrak{M}}$. Note that at this level of description, we cannot expressly rule out *Zeno* trajectories $\langle \delta_i, q_i, \gamma_i \rangle_{i \in I}$ such that I is infinite but $\sum_{i \in I} \delta_i < \infty$, but by considering variant evolution relations $\hat{e}_q$ defined using a minimal time duration δ , we could.

A clean μ -calculus definition of the higher-order modalities $\langle \mathbf{h} \rangle$ and $[\mathbf{h}]$ also opens up new possibilities for *aggregation* in complex systems. We could model a complex system as a hybrid “meta-automaton”, where the dynamics at each discrete meta-mode $p \in P$ are given by the reachability relation h_p of a (basic) hybrid automaton over state space $X_p \subseteq Q_p \times \mathbb{R}^n$, with switching relations from X_p to $X_{p'}$ between automata, as illustrated in Figure 2. We now have the machinery with which to formally reason about the dynamics of such a creature.

We also gain a clearer view of the enterprise of symbolic model checking for hybrid and real-time systems, as implemented in tools such as HYTECH and KRONOS. The basic task of such systems is to compute the *reachable region* of a hybrid dynamical system under the control of a given hybrid automaton $\mathcal{H}$. As noted in the recent paper of Henzinger, Kupferman and Qadeer [20], to capture the notion “reachable from φ ”, as distinct from “reaches φ ”, one needs in the semantics the *post-image*, rather than the pre-image, operator of a relation. The cleanest way to do it is to use the basic identity: $\text{post}[r] = \text{pre}[\check{r}]$, where $\check{r}$ is the relational *converse* or *inverse* of r , and to extend the μ -calculus with a converse operation governed by the rule:

$$\langle \check{a} \rangle \psi \rightarrow \varphi \quad \text{iff} \quad \psi \rightarrow [a]\varphi \quad (9)$$

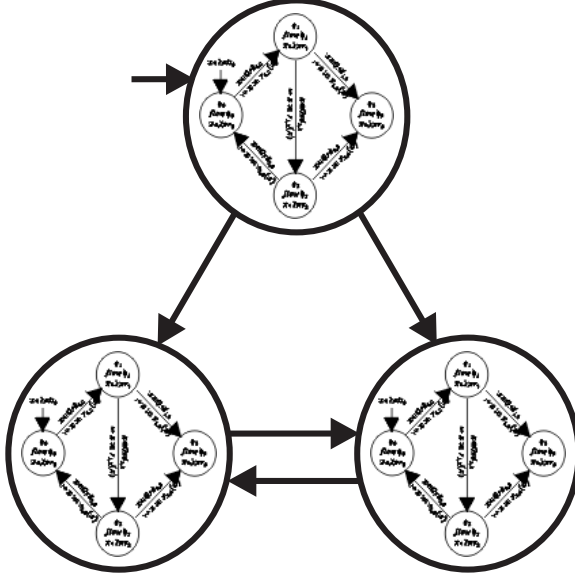


Fig. 2. Aggregation in complex systems

Then the sentence

$$\langle \tilde{\mathbf{h}} \rangle \mathbf{Init} \quad (10)$$

denotes the reachable region, where the *post* modalities $\langle \tilde{\mathbf{h}} \rangle$ and $[\tilde{\mathbf{h}}]$ are defined as in (5), but substituting the converse relations. Symbolic model checking tools

attempt to compute the *value* of $\|\langle \tilde{\mathbf{h}} \rangle \mathbf{Init}\|^\mathfrak{M}$ as a *first-order formula* in $n + 1$ free variables $(z, x_1, \dots, x_n)$, in the language $\mathcal{L}(\overline{\mathbb{R}})$ of, say, the structure $\overline{\mathbb{R}} = (\mathbb{R}; <, +, -, \cdot, 0, 1, \{\bar{q}\}_{q \in Q})$ as the real closed field² plus discrete constants. The procedure computes a sequence of first-order formulas $\chi_0, \chi_1, \dots, \chi_k, \dots$ which are *translations* of the μ -calculus formulas forming the approximation sequence for

$\langle \tilde{\mathbf{h}} \rangle \mathbf{Init}$, with the translation starting from the explicit first-order definitions of the set *Init* and the relations e_q and $c_{q,q'}$. The procedure terminates at stage $k + 1$ if the formula: $\chi_{k+1} \leftrightarrow \chi_k$ is provable in the first-order theory $Th(\overline{\mathbb{R}})$ of the relevant structure over $\mathbb{R}$, in which case the reachable region is defined by χ_k . The procedure is guaranteed to terminate when the model $\mathfrak{M} = \mathfrak{M}_\mathcal{H}$ has a *finite bisimulation quotient* $\mathfrak{M}^\approx$, where $\approx$ is an equivalence relation on $X \subseteq Q \times \mathbb{R}^n$ which *respects* each of the transition relations e_q and $c_{q,q'}$ and the

² The real closed field $\overline{\mathbb{R}}$ admits elimination of quantifiers, so all first-order formulas in the language are provably equivalent in the theory $Th(\overline{\mathbb{R}})$ to a *quantifier-free* formula. The definable subsets of $\mathbb{R}^n$ in $\overline{\mathbb{R}}$ are the *semi-algebraic* sets: finite unions of sets defined by equalities and inequalities over polynomials $f \in \mathbb{R}[X_1, \dots, X_n]$ [14].

observation sets $Init_q$, Inv_q , $Grd_{q,q'}$. The recent work by Lafferriere, Pappas, Sastry and Yovine [27], [28], identifies a class of systems whose LTS models $\mathfrak{M}_{\mathcal{H}}$ are first-order definable in an *o-minimal structure* $\overline{\mathbb{R}}$ expanding the real-closed field. The finite cell decomposition property of such structures (together with a restriction on the form of the controlled jumps relations $c_{q,q'}$) is used to construct the finite bisimulation equivalence. (The theory of definable sets in o-minimal structures is developed in van den Dries' monograph *Tame Topology and O-minimal Structures* [14].)

The basic propositional modal μ -calculus can provide both a usable and a richly expressive formalism for reasoning about the *abstract dynamics* of hybrid systems. We want and need more. We want to be able to express in our logical formalisms what we *mean* by *continuous* and *discrete* dynamics, and hybrids of the two. We want to be able to formally express notions of *imprecision* or *metric tolerance*, such as the property of “being within distance ϵ ” of a set, for a particular $\epsilon > 0$. More generally, we want a logical formalism that supports not only the specification and verification of single properties, but the larger task of representing and building up a *knowledge base* of properties of a system, starting with structural properties assumed in the modeling, and then adding new facts as they are verified by either model-checking or deductive means.

The remainder of this paper is an exploration of how the propositional modal μ -calculus can form a basis for a cohesive and expressively rich logical framework for the formal analysis of hybrid systems. In developing the logics, our key resources include:

1. *modal logics*, considered as a general formalism for reasoning about binary relations and operators on sets ([9], [35], [38], [5]); and
2. *set-valued analysis and dynamical systems theory*, brought into play by considering transition relations $r \subseteq X \times X$ in their equivalent form as *set-valued maps* $r : X \rightsquigarrow X$, i.e. functions $r : X \rightarrow \mathcal{P}(X)$ ([1], [6], [7]).

In the course of this paper, it will be important to keep an eye on both the distinction and the interplay between:

- the μ -calculus and various extensions as *propositional modal logics* (and thus ultimately *monadic second-order logics* [25]), in which formulas of the *same* formal language can be meaningfully interpreted in a variety of LTS models of any cardinality; in particular, in both continuum-sized models $\mathfrak{M}$ and in finite quotients $\mathfrak{M}^{\approx}$; and
- the *first-order* languages $\mathcal{L}(\overline{\mathbb{R}})$ and theories $Th(\overline{\mathbb{R}})$ of specific structures $\overline{\mathbb{R}} = (\mathbb{R}; <, +, -, \cdot, 0, 1, \dots)$ over the reals, used in defining the components – the state space X , the transition relations $a^{\mathfrak{M}}$ and observation sets $\|p\|^{\mathfrak{M}}$ – of particular, albeit intended, LTS models $\mathfrak{M}$.

With regard to the latter, note that in the theory of o-minimal structures, relations $r : \mathbb{R}^m \rightsquigarrow \mathbb{R}^n$ go by the name of *definable families* $(r_x)_{x \in \mathbb{R}^m}$ ([14] §3.3).

To restate the point, the *system description* language is that of first-order logic, while the *system specification* language is that of propositional polymodal logic with fixed-point quantifiers.

This paper is one installment of a larger project. An analysis of the concept of bisimulation, and its relation to the algebraic semantics for the μ -calculus, is given in [11], and [12] gives the completeness of deductive proof systems for normal polymodal extensions of the μ -calculus. Related logics and earlier versions of some of the ideas are found in [10].

The paper is organized as follows. Section 2 is a review and analysis of basic hybrid systems and their associated LTS models. Section 3 is a review of the syntax and LTS semantics of the modal μ -calculus. In Section 4, we flesh out the skeleton of an LTS model by imbuing the state space with topological and metric tolerance structure; we explore continuity and tolerance properties of relations $r : X \rightsquigarrow Y$ and applications to components of hybrid automata. Section 5 presents deductive proof systems for the new logics, extending Kozen's axiomatization of L_μ . Section 6 is a brief discussion of ongoing research.

2 Basic hybrid automata and associated LTS models

First, a note on notation. For a set X , $\mathcal{P}(X)$ denotes the family of all subsets of X (a complete Boolean algebra). Following [6], the notation $r : X \rightsquigarrow Y$ means $r \subseteq X \times Y$ is a relation, or equivalently, $r : X \rightarrow \mathcal{P}(Y)$ is a set-valued map, with values $r(x) \subseteq Y$ for $x \in X$. The expressions:

$$x \xrightarrow{r} y, \quad (x, y) \in r, \quad y \in r(x) \quad \text{and} \quad x r y$$

are synonymous. The *domain* of $r : X \rightsquigarrow Y$ is defined by $\text{dom}(r) \triangleq \sigma(r)(Y)$, and the *range* $\text{ran}(r) \triangleq \sigma(\tilde{r})(X) = \text{dom}(\tilde{r})$. Relational compositions $r \cdot s$ of $r : X \rightsquigarrow Y$ and $s : Y \rightsquigarrow Z$ are read from left to right in sequential order, defined by:

$$x \xrightarrow{r \cdot s} z \quad \triangleq \quad (\exists y \in Y) \quad x \xrightarrow{r} y \quad \text{and} \quad y \xrightarrow{s} z$$

(cf. [1] where composition is written in the reverse order, as for functional composition.)

We base our discussion on a generalization of the systems considered in [27],[28], depicted in Figure 1. Figure 3 is an illustration.

Definition 1. A (*basic, evolution time-deterministic*) hybrid system is a structure

$$\mathcal{H} = (Q, G, \{X_q\}_{q \in Q}, \{\phi_q\}_{q \in Q}, \{Init_q\}_{q \in Q}, \{Inv_q\}_{q \in Q}, \{r_{q,q'}\}_{(q,q') \in G}, \{Grd_{q,q'}\}_{(q,q') \in G})$$

where

- Q is a finite set of discrete states or control modes;
- $G \subseteq Q \times Q$ is the control graph of discrete transitions;
- for each $q \in Q$,
 - $X_q \subseteq \mathbb{R}^n$ is the state space for mode q ;
 - $\phi_q : X_q \times \mathbb{R}^+ \rightarrow X_q$ is the continuous semi-flow of a vector field on X_q ;
 - $Inv_q \subseteq X_q$ is the set of invariant states for mode q , or the domain of permitted evolution within mode q ;
 - $Init_q \subseteq Inv_q$ is the set of initial states for mode q (possibly empty);
- for each discrete transition $(q, q') \in G$,
 - $Grd_{q,q'} \subseteq X_q$ is the guard set for the jump from q to q' ;
 - $r_{q,q'} : X_q \rightsquigarrow X_{q'}$ is the reset relation;
 for $x \in X_q$, $r_{q,q'}(x) \subseteq X_{q'}$ is the set of possible reassignment states after the jump from q to q' .

The *hybrid state space* of the system $\mathcal{H}$ is the set

$$X = \bigcup_{q \in Q} \{q\} \times X_q$$

To keep things simple, assume a fixed number n of real-valued coordinates, so $X_q \subseteq \mathbb{R}^n$ for each $q \in Q$. In [27],[28], the systems under consideration are simpler again in that they have *constant* reset relations $r_{q,q'} = Grd_{q,q'} \times Rst_{q,q'}$, with the constant set of reassignment states $Rst_{q,q'} \subseteq Inv_{q'}$.

The intention is that a hybrid system, so defined, is the *semantic content* of a hybrid automaton in the sense of Henzinger [19], Def. 1.1. For definiteness, we take a (basic, evolution time-deterministic) *hybrid automata* to be a hybrid system $\mathcal{H}$ with a *concrete syntactic description*, namely:

- the discrete structure is given by a finite graph (Q, G) , where $G \subseteq Q \times Q$;
- each of the component sets X_q , $Init_q$, Inv_q , $Grd_{q,q'} \subseteq \mathbb{R}^n$, semi-flows $\phi_q : X_q \times \mathbb{R}^+ \rightarrow X_q$, and reset relations $r_{q,q'} \subseteq X_q \times X_{q'}$ have explicit first-order definitions in the language $\mathcal{L}(<, +, -, \cdot, 0, 1, \dots)$ of some specified structure $\overline{\mathbb{R}}$ over the reals.

From [27], [28], we have reason to want such a structure $\overline{\mathbb{R}}$ to be o-minimal.

Operationally, a hybrid automaton $\mathcal{H}$ can be thought of as defining a non-deterministic *hybrid control policy*, partially defined on states $(z, x) \in X$:

if $z = q$ and $x \in Inv_q$
then stay in discrete mode q and continue evolution according to ϕ_q ;
if $z = q$ and $x \in Grd_{q,q'}$ for some $(q, q') \in G$,
then switch to discrete mode q' , re-initialize to some $x' \in r_{q,q'}(x)$,
 and then evolve according to the flow $\phi_{q'}$.

The *domain of definition* of $\mathcal{H}$ is given by:

$$\text{dom}(\mathcal{H}) \doteq \left(\bigcup_{q \in Q} \{q\} \times Inv_q \right) \cup \left(\bigcup_{(q,q') \in G} \{q\} \times Grd_{q,q'} \right)$$

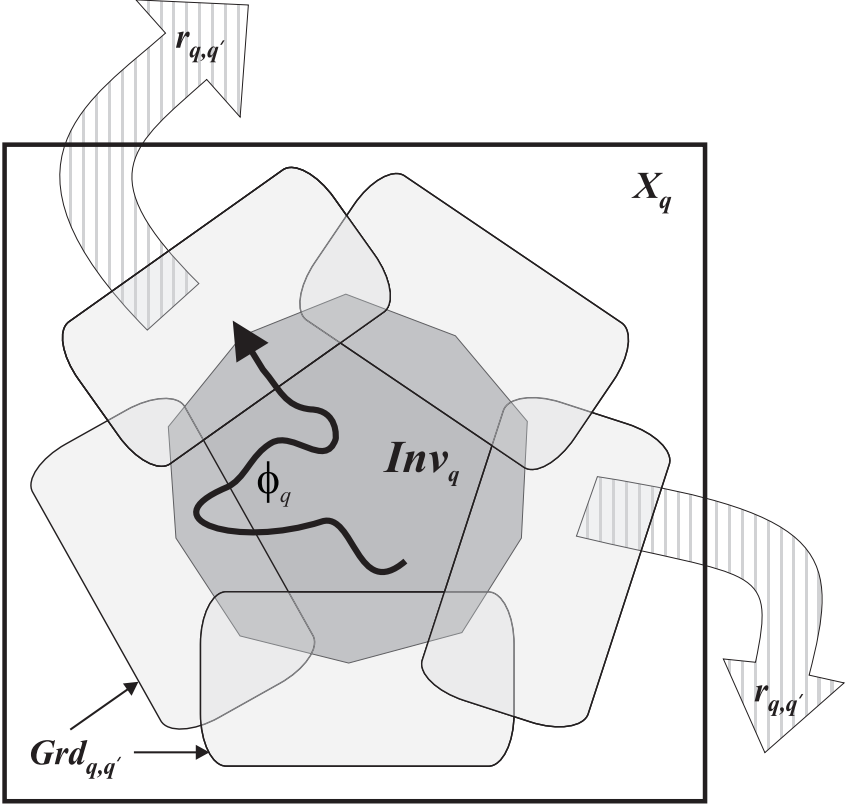


Fig. 3. Operation of basic hybrid automaton

If $z = q$ and $x \in Grd_{q,q'}$ for some $(q, q') \in G$, then that discrete control switch is said to be *enabled*; if $(q, x) \in \text{dom}(\mathcal{H})$ but $x \notin Inv_q$, then some discrete control switch is said to be *forced*. It is generally assumed that $r_{q,q'}(x) \subseteq Inv_{q'}$ for all $x \in Grd_{q,q'}$; in words, $Inv_{q'}$ is (forward) $r_{q,q'}$ -invariant from $Grd_{q,q'}$. In some expositions (e.g. [27]), it is required that $\mathcal{H}$ be *total* or *non-blocking*, which amounts to the assumption that $\text{dom}(\mathcal{H}) = X$.

In descriptions of the operation of a hybrid automaton and the ensuing class of trajectories of the system, it is generally assumed (e.g. [19]) that the state $x = (x_1, \dots, x_n) \in \mathbb{R}^n$ of the physical plant is being *continuously sensed*, with *perfect precision*, and that the action and effect of a discrete control switch is *instantaneous*.

The accepted ([19], [27]) definition of the (“time-abstract”) transition system of a hybrid automaton, with modified notation, is as follows.

Definition 2. *Given a hybrid system $\mathcal{H}$, the LTS model $\mathfrak{M}_{\mathcal{H}}$ determined by $\mathcal{H}$ has the following components:*

- the state space $X \doteq \cup_{q \in Q} \{q\} \times X_q$;

- for each discrete state $q \in Q$, the constrained evolution relation $e_q : X_q \rightsquigarrow X_q$ defined by:

$$x \xrightarrow{e_q} x' \quad \doteq \quad (\exists t \in \mathbb{R}^+) [x' = \phi_q(x, t) \wedge (\forall s \in [0, t]) \phi_q(x, s) \in \text{Inv}_q]$$

- for each discrete transition $(q, q') \in G$, the controlled jump relation $c_{q,q'} : X_q \rightsquigarrow X_{q'}$ defined by:

$$x \xrightarrow{c_{q,q'}} x' \quad \doteq \quad x \in \text{Grd}_{q,q'} \wedge x' \in \text{Inv}_{q'} \wedge x \xrightarrow{r_{q,q'}} x'$$

- the observation sets $X_q, \text{Init}_q, \text{Inv}_q, \text{Grd}_{q,q'}$.

We adopt the notational convention of identifying, when convenient, sets $A_q \subseteq X_q$ and $\{q\} \times A_q \subseteq X$; moreover, the relations $e_q : X_q \rightsquigarrow X_q$ and $c_{q,q'} : X_q \rightsquigarrow X_{q'}$ can be “lifted” to relations $X \rightsquigarrow X$ in the obvious way.

From the definition of the evolution relation e_q , a desired property of the domain of evolution Inv_q is that it be *convex* with respect to the semi-flow ϕ_q , in the sense that:

$$\begin{aligned} &\text{if } x \in \text{Inv}_q \text{ and } \phi_q(x, t) \in \text{Inv}_q \text{ for some } t \geq 0, \\ &\text{then } \phi_q(x, s) \in \text{Inv}_q \text{ for all } s \in [0, t] \end{aligned}$$

So no curve segment of the semi-flow with both endpoints in Inv_q ever leaves Inv_q at an intermediate point.

In the terminology of [1] Ch. 6, Definition 6.3, the (positive) *orbit relation* $f : X \rightsquigarrow X$ of a semi-flow $\phi : X \times \mathbb{R}^+ \rightarrow X$ is defined by:

$$x \xrightarrow{f} x' \quad \doteq \quad (\exists t \in \mathbb{R}^+) \ x' = \phi(x, t) \quad (11)$$

With respect to the orbit relation $f_q : X_q \rightsquigarrow X_q$ of ϕ_q , the desired convexity property for Inv_q has the form:

$$\text{if } x_0, x_1 \in \text{Inv}_q \text{ and } x_0 \xrightarrow{f_q} x \xrightarrow{f_q} x_1 \text{ then } x \in \text{Inv}_q$$

So when Inv_q is f_q -convex, we have the decompositions

$$e_q = f_q \cap (\text{Inv}_q \times \text{Inv}_q) \quad \text{and} \quad c_{q,q'} = r_{q,q'} \cap (\text{Grd}_{q,q'} \times \text{Inv}_{q'})$$

in which case we may as well assume the LTS model $\mathfrak{M}_{\mathcal{H}}$ includes the (unconstrained) orbit relations f_q and the uncontrolled reset relation $r_{q,q'}$. If we want to express properties which require *both* the orbit relation f_q and its converse

(convexity is one such), then we should include $\widetilde{f_q}$ as a component of $\mathfrak{M}_{\mathcal{H}}$ as well (see also [20]).

The modularity of the modal μ -calculus allows us to succinctly express not only *desired properties* – i.e. those to be verified, but also various of the structural properties of the LTS model $\mathfrak{M}_{\mathcal{H}}$ that it will typically possess *by assumption*. In

a deductive framework, such sentences and sentence schemes (formulas with free propositional variables Z) provide an initial stock of facts known to be true in the model, and serve as hypotheses in application of inference rules when seeking to expand one's stock of knowledge.

- [1] $\langle \widetilde{\mathbf{f}}_q \rangle \mathbf{Inv}_q \wedge \langle \mathbf{f}_q \rangle \mathbf{Inv}_q \rightarrow \mathbf{Inv}_q$
- [2] $\mathbf{Init}_q \rightarrow \mathbf{Inv}_q$
- [3] $\mathbf{Init} \leftrightarrow \bigvee_{q \in Q} \mathbf{Init}_q$
- [4] $\mathbf{Inv} \leftrightarrow \bigvee_{q \in Q} \mathbf{Inv}_q$
- [5] $\langle \widetilde{\mathbf{r}}_{q,q'} \rangle \mathbf{Grd}_{q,q'} \rightarrow \mathbf{Inv}_{q'}$
- [6] $\mathbf{Grd}_{q,q'} \rightarrow \langle \mathbf{r}_{q,q'} \rangle \mathbf{tt}$
- [7] $\langle \mathbf{e}_q \rangle Z \leftrightarrow \mathbf{Inv}_q \wedge \langle \mathbf{f}_q \rangle (Z \wedge \mathbf{Inv}_q)$
- [8] $\langle \widetilde{\mathbf{e}}_q \rangle Z \leftrightarrow \mathbf{Inv}_q \wedge \langle \widetilde{\mathbf{f}}_q \rangle (Z \wedge \mathbf{Inv}_q)$
- [9] $\langle \mathbf{c}_{q,q'} \rangle Z \leftrightarrow \mathbf{Grd}_{q,q'} \wedge \langle \mathbf{r}_{q,q'} \rangle (Z \wedge \mathbf{Inv}_{q'})$
- [10] $\langle \widetilde{\mathbf{c}}_{q,q'} \rangle Z \leftrightarrow \mathbf{Inv}_{q'} \wedge \langle \widetilde{\mathbf{r}}_{q,q'} \rangle (Z \wedge \mathbf{Grd}_{q,q'})$
- [11] $\langle \mathbf{f} \rangle Z \leftrightarrow \bigvee_{q \in Q} \langle \mathbf{f}_q \rangle Z$
- [12] $Z \rightarrow \langle \mathbf{f} \rangle Z$
- [13] $\langle \mathbf{f}_q \rangle \langle \mathbf{f}_q \rangle Z \rightarrow \langle \mathbf{f}_q \rangle Z$
- [14] $\langle \mathbf{e} \rangle Z \leftrightarrow \bigvee_{q \in Q} \langle \mathbf{e}_q \rangle Z$
- [15] $\langle \mathbf{c} \rangle Z \leftrightarrow \bigvee_{(q,q') \in G} \langle \mathbf{c}_{q,q'} \rangle Z$
- [16] $\langle \mathbf{h} \rangle \mathbf{tt} \leftrightarrow \bigvee_{q \in Q} \mathbf{Inv}_q \vee \bigvee_{(q,q') \in G} \mathbf{Grd}_{q,q'}$

[1] says that $\mathbf{Inv}_q$ is f_q -convex. [2] is merely that $\mathbf{Init}_q \subseteq \mathbf{Inv}_q$. [3] and [4] define the global initial and invariant sets. [5] is the assumption that $\mathbf{Inv}_{q'}$ is (future) $r_{q,q'}$ -invariant from $\mathbf{Grd}_{q,q'}$. [6] says that every point in $\mathbf{Grd}_{q,q'}$ has an $r_{q,q'}$ -successor; i.e. $\mathbf{Grd}_{q,q'} \subseteq \text{dom}(r_{q,q'})$. [7] – [10] follow from the decompositions $e_q = f_q \cap (\mathbf{Inv}_q \times \mathbf{Inv}_q)$ and $c_{q,q'} = r_{q,q'} \cap (\mathbf{Grd}_{q,q'} \times \mathbf{Inv}_{q'})$. In particular, using the rule for converse (9) in Section 1 above, we have:

$$\varphi \rightarrow [\mathbf{e}_q] \varphi \quad \text{iff} \quad \mathbf{Inv}_q \wedge \langle \widetilde{\mathbf{f}}_q \rangle (\varphi \wedge \mathbf{Inv}_q) \rightarrow \varphi \quad (12)$$

and

$$\varphi \rightarrow [\mathbf{c}_{q,q'}] \varphi \quad \text{iff} \quad \mathbf{Inv}_{q'} \wedge \langle \widetilde{\mathbf{r}}_{q,q'} \rangle (\varphi \wedge \mathbf{Grd}_{q,q'}) \rightarrow \varphi \quad (13)$$

[11] defines f as the union of the orbit relations f_q . From the zero semi-flow property, each f_q is reflexive on its domain X_q , so f is reflexive (and total) on the whole space X , which is [12]. From the sum semi-flow property, each f_q is transitive; this is [13]. [14] and [15] are the definitions $e \triangleq \cup_{q \in Q} e_q$ and $c \triangleq \cup_{(q,q') \in G} c_{q,q'}$. From [7], [14] and [12], it follows that:

$$(Z \wedge \mathbf{Inv}) \rightarrow \langle \mathbf{e} \rangle (Z \wedge \mathbf{Inv}) \quad (14)$$

that is, the relational sum e is reflexive on its domain. And from [7] and [13], we get:

$$\langle \mathbf{e}_q \rangle \langle \mathbf{e}_q \rangle Z \rightarrow \langle \mathbf{e}_q \rangle Z \quad (15)$$

which says each e_q is transitive.

[16] defines the domain $\text{dom}(\mathcal{H})$. The definitions of $\langle \mathbf{h} \rangle$ and $[\mathbf{h}]$ in (5) above should also be added to the list.

Using convexity assumption [1] and (12), the invariance assumption [5] and (13), and the invariance rule (7), it follows that $\mathbf{Inv} \rightarrow [\mathbf{h}] \mathbf{Inv}$ will be true in $\mathfrak{M}_{\mathcal{H}}$; i.e. the set Inv is future-invariant under $\mathcal{H}$. More generally, whenever $\mathbf{Inv} \rightarrow \varphi$ is true in $\mathfrak{M}_{\mathcal{H}}$, then $\mathbf{Init} \rightarrow [\mathbf{h}] \varphi$ will be true, and thus on the current interpretation, $\|\varphi\|^{\mathfrak{M}}$ is *safe* under the action of $\mathcal{H}$, since no (perfect precision) hybrid trajectory starting in *Init* ever leaves *Inv*. So in this scenario, the situation of a controlled jump being *forced* – that is, $(q, x) \in \text{dom}(\mathcal{H})$ but $x \notin \text{Inv}_q$ – can in fact never arise. Perfect precision trajectories start or land inside Inv_q , evolve continuously according to ϕ_q , and then while the state is still *inside* Inv_q , or at worst on the (topological) *boundary* of Inv_q , a jump is made according to $c_{q,q'}$.

In some accounts of the LTS model of a hybrid automata (including that in [19]), the definition of the constrained evolution relation e_q is slightly weaker, with the requirement: $\forall s \in [0, t], \phi_q(x, s) \in \text{Inv}_q$, so the end-point $\phi_q(x, t)$ need not lie in Inv_q . If Inv_q is *closed* (in the standard topology on $X_q \subseteq \mathbb{R}^n$), then the continuity of $\phi_q : X_q \times \mathbb{R}^+ \rightarrow X_q$ entails that all such end-points *will* lie in Inv_q regardless, so the weakening makes no difference. In virtually all concrete examples of hybrid automata in the literature, the invariant sets Inv_q are closed.

In Section 4, when we adjoin modalities corresponding to the interior and closure operators of a topology, we will be able to formally express properties such as being open, closed, or the topological boundary of a set. We will also be able to give formal expression to the assumption that the orbit relations f_q are those of *continuous* semi-flows, and to consider consequences of continuity.

We also clearly need to entertain the possibility that a physical realization of a hybrid automaton as a control policy might be *less than perfect*: sensors will be accurate only up to some level of precision; we should allow for delay between sensing the state and acting on that sensor reading in accordance with the control policy; and then there are margins of error in real-valued constants used in first-order definitions of the components of the model. In Section 4, we will consider alternative classes of hybrid trajectories by playing with the definitions of the fixed-point modalities $\langle \mathbf{h} \rangle$ and $[\mathbf{h}]$ in an enriched modal language containing

modalities $\langle \epsilon \rangle$ and $[\epsilon]$ interpreted by metric ϵ -tolerance relations, for concrete values of $\epsilon > 0$.

3 Syntax and LTS semantics of the modal μ -calculus

The μ -calculus originated in the late 1960's (Scott and de Bakker) as a formal logic of digital programs, the input-output behavior of an atomic program being represented as a binary transition relation on (discrete) states. Contemporary introductions to the μ -calculus can be found in [38], [15]. In this section, we review the syntax and semantics over LTS models of the propositional modal μ -calculus.

Definition 3. A modal signature is a pair (Φ, Σ) , where Φ is a set of propositional constants and Σ is a set of transition labels. Let PVar denote a fixed set of propositional (second-order or set-valued) variables. The collection $\mathcal{F}_\mu(\Phi, \Sigma)$ of formulas of the propositional modal μ -calculus is generated by the grammar:

$$\varphi ::= \mathbf{ff} \mid p \mid Z \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \langle a \rangle \varphi \mid \mu Z. \varphi$$

for propositional constants $p \in \Phi$, propositional variables $Z \in \text{PVar}$, and transition labels $a \in \Sigma$, and with the proviso that in $\mu Z. \varphi$, the variable Z occur positively, i.e. each occurrence of Z in φ is within the scope of an even number of negations.

The other (classical) propositional connectives, modalities and greatest fixed point quantifier are defined in the usual way:

$$\begin{aligned} \mathbf{tt} &\doteq \neg \mathbf{ff} & \varphi_1 \wedge \varphi_2 &\doteq \neg(\neg\varphi_1 \vee \neg\varphi_2) \\ \varphi_1 \rightarrow \varphi_2 &\doteq \neg\varphi_1 \vee \varphi_2 & \varphi_1 \leftrightarrow \varphi_2 &\doteq (\varphi_1 \rightarrow \varphi_2) \wedge (\varphi_2 \rightarrow \varphi_1) \\ [a]\varphi &\doteq \neg\langle a \rangle \neg\varphi & \nu Z. \varphi &\doteq \neg\mu Z. \neg\varphi[Z := \neg Z] \end{aligned}$$

An occurrence of a variable $Z \in \text{PVar}$ in a formula that is within the scope of a μZ is called *bound*, otherwise it is *free* (as in first-order logic). Let $\mathcal{S}_\mu(\Phi, \Sigma)$ denote the set of all *sentences*, or closed formulas of $\mathcal{F}_\mu(\Phi, \Sigma)$, i.e. those without any free variables, and let $\mathcal{F}(\Phi, \Sigma)$ and $\mathcal{S}(\Phi, \Sigma)$ denote, respectively, the set of all purely *modal* formulas and sentences, i.e. those containing no fixed point quantifiers, and in case of sentences, no variables Z .

For formulas $\varphi, \psi \in \mathcal{F}_\mu(\Phi, \Sigma)$, let $\varphi[Z := \psi]$ denote the result substituting ψ for all free occurrences of Z . By renaming bound variables in φ if necessary, we can assume such substitutions do not result in the unintended capture of free variables.

Definition 4. Given an LTS $\mathfrak{M} = (X, \{a^\mathfrak{M}\}_{a \in \Sigma}, \{\|p\|^\mathfrak{M}\}_{p \in \Phi})$ of modal signature (Φ, Σ) , a (propositional, or second-order) variable assignment in $\mathfrak{M}$ is any

map $\xi : \text{PVar} \rightarrow \mathcal{P}(X)$. Each such assignment ξ uniquely extends to a denotation map $\|\cdot\|_\xi^{\mathfrak{M}} : \mathcal{F}_\mu(\Phi, \Sigma) \rightarrow \mathcal{P}(X)$ as follows:

$$\begin{aligned}
\|\mathbf{ff}\|_\xi^{\mathfrak{M}} &\doteq \emptyset \\
\|p\|_\xi^{\mathfrak{M}} &\doteq \|p\|^{\mathfrak{M}} && \text{for } p \in \Phi \\
\|Z\|_\xi^{\mathfrak{M}} &\doteq \xi(Z) && \text{for } Z \in \text{PVar} \\
\|\neg\varphi\|_\xi^{\mathfrak{M}} &\doteq X - \|\varphi\|_\xi^{\mathfrak{M}} \\
\|\varphi_1 \vee \varphi_2\|_\xi^{\mathfrak{M}} &\doteq \|\varphi_1\|_\xi^{\mathfrak{M}} \cup \|\varphi_2\|_\xi^{\mathfrak{M}} \\
\|\langle a \rangle \varphi\|_\xi^{\mathfrak{M}} &\doteq \sigma(a^{\mathfrak{M}}) \left(\|\varphi\|_\xi^{\mathfrak{M}} \right) && \text{for } a \in \Sigma \\
\|\mu Z. \varphi\|_\xi^{\mathfrak{M}} &\doteq \bigcap \{ A \in \mathcal{P}(X) \mid \|\varphi\|_{\xi(A/Z)}^{\mathfrak{M}} \subseteq A \}
\end{aligned}$$

where the pre-image operator $\sigma(a^{\mathfrak{M}})$ is defined as in (2) above, and for sets $A \in \mathcal{P}(X)$, the variant assignment $\xi(A/Z) : \text{PVar} \rightarrow \mathcal{P}(X)$ is given by:

$$\xi(A/Z)(W) = \xi(W) \text{ if } W \neq Z, \quad \text{and} \quad \xi(A/Z)(W) = A \text{ if } W = Z.$$

For formulas $\varphi \in \mathcal{F}_\mu(\Phi, \Sigma)$ and assignments $\xi : \text{PVar} \rightarrow \mathcal{P}(X)$ in $\mathfrak{M}$, we say:

- φ is true at state x in $(\mathfrak{M}, \xi)$, written: $\mathfrak{M}, \xi, x \models \varphi$, iff $x \in \|\varphi\|_\xi^{\mathfrak{M}}$;
- φ is true in $(\mathfrak{M}, \xi)$, written: $\mathfrak{M}, \xi \models \varphi$, iff $\|\varphi\|_\xi^{\mathfrak{M}} = X$; i.e. φ is true at all states x in $(\mathfrak{M}, \xi)$; and
- φ is true in $\mathfrak{M}$, written: $\mathfrak{M} \models \varphi$, iff φ is true in $(\mathfrak{M}, \xi)$ for all assignments ξ in $\mathfrak{M}$.

For sentences $\varphi \in \mathcal{S}_\mu(\Phi, \Sigma)$, the denotation $\|\varphi\|_\xi^{\mathfrak{M}}$ is independent of the variable assignment ξ , and is written $\|\varphi\|^{\mathfrak{M}}$. So $\mathfrak{M} \models \varphi$ iff $\mathfrak{M}, \xi \models \varphi$ for any assignment ξ .

Given a model $\mathfrak{M}$ and variable assignment ξ , each formula $\varphi \in \mathcal{F}_\mu(\Phi, \Sigma)$ and each variable $Z \in \text{PVar}$ free in φ , together determine an operator on sets $\varphi_{\xi, Z}^{\mathfrak{M}} : \mathcal{P}(X) \rightarrow \mathcal{P}(X)$ given by:

$$(\varphi_{\xi, Z}^{\mathfrak{M}})(A) \doteq \|\varphi\|_{\xi(A/Z)}^{\mathfrak{M}} \quad (16)$$

The variant assignment construct corresponds to substitution: for all formulas $\psi \in \mathcal{F}_\mu(\Phi, \Sigma)$,

$$(\varphi_{\xi, Z}^{\mathfrak{M}})(\|\psi\|_\xi^{\mathfrak{M}}) = \|\varphi[Z := \psi]\|_\xi^{\mathfrak{M}} \quad (17)$$

When the variable Z occurs positively within φ , so $\mu Z. \varphi \in \mathcal{F}_\mu(\Phi, \Sigma)$, the operator $\varphi_{\xi, Z}^{\mathfrak{M}}$ is $\subseteq$ -monotone:

$$A \subseteq B \quad \Rightarrow \quad F(A) \subseteq F(B)$$

for $F = \varphi_{\xi, Z}^{\mathfrak{M}}$. The clause in Definition 4 for μ -formulas says that $\|\mu Z.\varphi\|_{\xi}^{\mathfrak{M}}$ is the $\subseteq$ -least pre-fixed-point of the monotone operator $\varphi_{\xi, Z}^{\mathfrak{M}}$ in the complete lattice $\mathcal{P}(X)$. So by the Tarski-Knaster fixed-point theorem, $\|\mu Z.\varphi\|_{\xi}^{\mathfrak{M}}$ must also be the $\subseteq$ -least fixed-point of $\varphi_{\xi, Z}^{\mathfrak{M}}$; that is:

$$\|\mu Z.\varphi\|_{\xi}^{\mathfrak{M}} = \bigcap \{A \in \mathcal{P}(X) \mid \|\varphi\|_{\xi(A/Z)}^{\mathfrak{M}} = A\}$$

In the standard set-theoretic semantics for the μ -calculus, as presented here and given in [23], [38], [40], [15], the propositional variables Z range over the full power-set (and complete Boolean algebra) $\mathcal{P}(X)$ – that is, *all* subsets of X . An alternative, developed by Kwiatkowska and colleagues [5], [8], is an *algebraic semantics* in which the range of propositional variables is restricted to a *subfamily* $\mathcal{A} \subseteq \mathcal{P}(X)$. This work has roots in a number of classic studies from the 1950's, notably that of Henkin [18] on completeness of higher-order logic; of Jónsson and Tarski [26] on Boolean algebras with operators; and that of Rasiowa and Sikorski [36] on algebraic logic.

Definition 5. ([5], [8]). *Given an LTS model $\mathfrak{M}$, a family of sets $\mathcal{A} \subseteq \mathcal{P}(X)$ is said to be a modal algebra for $\mathfrak{M}$, and the pair $(\mathfrak{M}, \mathcal{A})$ is known as a modal frame, when each of the following holds:*

1. $\mathcal{A}$ contains each of the observation sets $\|p\|^{\mathfrak{M}}$, for $p \in \Phi$;
2. $\mathcal{A}$ is a Boolean algebra under the finitary set-theoretic operations; and
3. $\mathcal{A}$ is closed under each of the pre-image operators $\sigma(a^{\mathfrak{M}})$ and $\tau(a^{\mathfrak{M}})$, for $a \in \Sigma$.

For purely modal formulas $\varphi \in \mathcal{F}(\Phi, \Sigma)$, the clauses in the inductive definition of the denotation $\|\varphi\|_{\xi}^{\mathcal{A}} \subseteq X$ with respect to a modal frame $(\mathfrak{M}, \mathcal{A})$ are identical to those in Definition 4 for $\|\varphi\|_{\xi}^{\mathfrak{M}}$, with the proviso that variable assignments ξ are restricted to $\mathcal{A}$, i.e. $\xi : \text{PVar} \rightarrow \mathcal{A}$.

A formula φ is true in the frame $(\mathfrak{M}, \mathcal{A})$, written $(\mathfrak{M}, \mathcal{A}) \models \varphi$, iff $\|\varphi\|_{\xi}^{\mathcal{A}} = X$ for all assignments ξ in $\mathcal{A}$.

An LTS model $\mathfrak{M}$ is identified with the modal frame $(\mathfrak{M}, \mathcal{P}(X))$.

Modal algebras $\mathcal{A} \subseteq \mathcal{P}(X)$ need not be complete as lattices, so unlike $\mathcal{P}(X)$, we have no guarantee that the set being the $\subseteq$ -least pre-fixed-point of $\varphi_{\xi, Z}^{\mathcal{A}}$ in fact exists in $\mathcal{A}$; when it does, it is the least fixed-point in $\mathcal{A}$ of $\varphi_{\xi, Z}^{\mathcal{A}}$, by a variant of the argument in the Tarski-Knaster fixed-point theorem.

Definition 6. ([5], [8]). *A modal algebra $\mathcal{A} \subseteq \mathcal{P}(X)$ is called a modal μ -algebra, and the pair $(\mathfrak{M}, \mathcal{A})$ called a modal μ -frame, if for each formula $\mu Z.\varphi \in \mathcal{F}_{\mu}(\Phi, \Sigma)$ the infinitary meet or infimum of the family in $\mathcal{A}$ of pre-fixed-points of $\varphi_{\xi, Z}^{\mathcal{A}}$*

$$\bigwedge \{A \in \mathcal{A} \mid \|\varphi\|_{\xi(A/Z)}^{\mathcal{A}} \subseteq A\}$$

exists in $\mathcal{A}$, in which case $\|\mu Z.\varphi\|_{\xi}^{\mathcal{A}}$ is that set.

In general, the denotations $\|\varphi\|_{\xi}^{\mathfrak{M}}$ and $\|\varphi\|_{\xi}^{\mathcal{A}}$ part company on μ -formulas, since the smallest of *all sets* $A \in \mathcal{P}(X)$ such that a condition holds will be contained in the smallest of all sets $A \in \mathcal{A}$ for which the same condition holds. In [11], we identify conditions under which a modal μ -frame $(\mathfrak{M}, \mathcal{A})$ is in *semantic agreement* with $\mathfrak{M}$, i.e. for all μ -formulas $\varphi \in \mathcal{F}_{\mu}(\Phi, \Sigma)$, $\|\varphi\|_{\xi}^{\mathcal{A}} = \|\varphi\|_{\xi}^{\mathfrak{M}}$ for all assignments ξ restricted to $\mathcal{A}$. The *smallest* μ -algebra for an LTS $\mathfrak{M}$ is the countable algebra

$$\mathcal{S}_{\mu}^{\mathfrak{M}} \doteq \{ \|\varphi\|_{\xi}^{\mathfrak{M}} \mid \varphi \in \mathcal{S}_{\mu}(\Phi, \Sigma) \}$$

of denotations of μ -sentences in $\mathfrak{M}$. It is readily verified that $\mathcal{S}_{\mu}^{\mathfrak{M}}$ is in semantic agreement $\mathfrak{M}$.

From the purely modal clauses in Definition 4, together with the definitions of the pre-image operators in (2), it follows that if the state space, transition relations and observation sets of an LTS model $\mathfrak{M}$ are all first-order definable in some structure, then for all *modal* sentences $\varphi \in \mathcal{S}(\Phi, \Sigma)$, the denotation $\|\varphi\|^{\mathfrak{M}} \subseteq X$ is first-order definable. Otherwise put, the countable algebra

$$\mathcal{S}^{\mathfrak{M}} \doteq \{ \|\varphi\|^{\mathfrak{M}} \mid \varphi \in \mathcal{S}(\Phi, \Sigma) \}$$

of denotations in $\mathfrak{M}$ of purely modal sentences, has a *finitary syntactic representation* as a family of first-order formulas; a family finitely generated by the explicit first-order definitions of the components of $\mathfrak{M}$, under the straight-forward translation of modal sentences based on the definitions (2) and the (classical) meaning of the Boolean connectives. Of course, an optimal situation is when the first-order structure admits *quantifier-elimination*, as then the naive translation of a modal sentence can be reduced to a quantifier-free formula, and so the algebra $\mathcal{S}^{\mathfrak{M}}$ will have a simpler and more tractable representation. Such algebras are the semantic content of Henzinger's notion of a *symbolic execution theory* in [19] §3.1.

Returning to the standard set-theoretic semantics, the completeness of $\mathcal{P}(X)$ as lattice ensures that the set $\|\mu Z.\varphi\|_{\xi}^{\mathfrak{M}}$ has an equivalent characterization (by the Park-Hitchcock fixed-point theorem) as the union of an $\subseteq$ -increasing sequence of approximations:

$$\|\mu Z.\varphi\|_{\xi}^{\mathfrak{M}} = \bigcup_{\alpha < \text{Ord}(\mathfrak{M})} \|\varphi\|_{\xi, \alpha}^{\mathfrak{M}}$$

where

$$\begin{aligned} \|\varphi\|_{\xi, 0}^{\mathfrak{M}} &\doteq \emptyset \\ \|\varphi\|_{\xi, \alpha+1}^{\mathfrak{M}} &\doteq \varphi_{\xi, Z}^{\mathfrak{M}} \left(\|\varphi\|_{\xi, \alpha}^{\mathfrak{M}} \right) \\ \|\varphi\|_{\xi, \eta}^{\mathfrak{M}} &\doteq \bigcup_{\alpha < \eta} \|\varphi\|_{\xi, \alpha}^{\mathfrak{M}} \quad \text{for limit ordinals } \eta \end{aligned}$$

and $Ord(\mathfrak{M}) < \kappa^+$, for $\kappa = Card(X)$, is the closure ordinal of $\mathfrak{M}$. The sets $\|\varphi\|_{\xi, \alpha}^{\mathfrak{M}}$ are μ -approximations of $\|\mu Z.\varphi\|_{\xi}^{\mathfrak{M}}$. Likewise, the denotation of $\nu Z.\varphi$ can be represented as the intersection of an $\subseteq$ -decreasing sequence of ν -approximations.

In the general case, over LTS models $\mathfrak{M}$ of arbitrary cardinality, approximation sequences for the denotation of fixed-point formulas proceed through transfinite ordinals; when X has the cardinality of the continuum, $Ord(\mathfrak{M})$ could be much longer than we care to deal with.

When the operator $\varphi_{\xi, Z}^{\mathfrak{M}}$ corresponding to the body of a μ -formula $\mu Z.\varphi$ is ω -chain-additive, that is, for $F = \varphi_{\xi, Z}^{\mathfrak{M}}$

$$F\left(\bigcup_{n < \omega} A_n\right) = \bigcup_{n < \omega} F(A_n) \quad \text{where } A_n \subseteq A_{n+1} \text{ for all } n < \omega$$

then the ordinal of convergence for $\|\mu Z.\varphi\|_{\xi}^{\mathfrak{M}}$ is at worst ω . In this case, we have a sequence of approximation formulas

$$\varphi^0 \doteq \text{ff} \quad \text{and} \quad \varphi^{n+1} \doteq \varphi[Z := \varphi^n] \quad \text{for } n < \omega \quad (18)$$

and

$$\|\mu Z.\varphi\|_{\xi}^{\mathfrak{M}} = \bigcup_{n < \omega} \|\varphi^n\|_{\xi}^{\mathfrak{M}}$$

since $\|\varphi^n\|_{\xi}^{\mathfrak{M}} = \|\varphi\|_{\xi, n}^{\mathfrak{M}}$. The terms “order-continuous” and “continuous from below” are also used instead of ω -chain-additive, since such an $F : \mathcal{P}(X) \rightarrow \mathcal{P}(X)$ is a continuous function with respect to the *Scott topology* on the complete partial order $(\mathcal{P}(X), \subseteq)$. We adapt the terminology of Jónsson and Tarski [26] on Boolean algebras with operators, since we are interested in other meanings of “continuous”. Dually, when $\varphi_{\xi, Z}^{\mathfrak{M}}$ is ω -chain-multiplicative, the ordinal of convergence for $\|\nu Z.\varphi\|_{\xi}^{\mathfrak{M}}$ is at worst ω , and the sequence of approximation formulas starts at **tt** and decreases.

In particular, the semantic operator corresponding to the body of $\langle \mathbf{h} \rangle \varphi$ (or $\langle \widetilde{\mathbf{h}} \rangle \varphi$), as defined in (5), for sentences φ , is:

$$A \longmapsto \sigma(e)(\|\varphi\|_{\xi}^{\mathfrak{M}}) \cup \sigma(ec)(A)$$

Since the $\exists$ -pre-image of any relation is *completely additive*, i.e. distributes over arbitrary unions, it follows that $\|\langle \mathbf{h} \rangle \varphi\|_{\xi}^{\mathfrak{M}}$ is the union of the denotations of the approximation sequence

$$\text{ff}, \quad \langle \mathbf{e} \rangle \varphi, \quad \langle \mathbf{e} \rangle \varphi \vee \langle \mathbf{e} \rangle \langle \mathbf{c} \rangle \langle \mathbf{e} \rangle \varphi, \quad \langle \mathbf{e} \rangle \varphi \vee \langle \mathbf{e} \rangle \langle \mathbf{c} \rangle \langle \mathbf{e} \rangle \varphi \vee \langle \mathbf{e} \rangle \langle \mathbf{c} \rangle \langle \mathbf{e} \rangle \langle \mathbf{c} \rangle \langle \mathbf{e} \rangle \varphi, \dots$$

Dually, the semantic operator corresponding to $[\mathbf{h}]$ is *completely multiplicative*.

When $\approx$ is a *bisimulation equivalence* on $\mathfrak{M}$ – that is, an equivalence relation on X which *respects* the transition relations $a^{\mathfrak{M}}$ and the observation sets $\|p\|_{\xi}^{\mathfrak{M}}$

in a suitable sense³ – then the fundamental property of truth-preservation is as follows: for all sentences $\varphi \in \mathcal{S}_\mu(\Phi, \Sigma)$ and all $x, y \in X$,

$$x \approx y \quad \Rightarrow \quad [x \in \|\varphi\|^\mathfrak{M} \Leftrightarrow y \in \|\varphi\|^\mathfrak{M}] \quad (19)$$

It follows that if $\approx$ is a bisimulation equivalence of *finite* index N , then the denotation $\|\varphi\|^\mathfrak{M}$ of each sentence is a *finite* union of equivalence classes under $\approx$. Hence for sentences $\mu Z.\varphi$ and $\nu Z.\varphi$, the ordinal of convergence for $\|\mu Z.\varphi\|^\mathfrak{M}$ and $\|\nu Z.\varphi\|^\mathfrak{M}$ is bounded by N . In this case, the finite quotient LTS $\mathfrak{M}^\approx$ is a finite simulacrum, and finite automaton representation, of the original system $\mathfrak{M}$. If such is the case, the countable μ -algebra $\mathcal{S}_\mu^\mathfrak{M}$ is in fact a *finite* algebra, and the atoms of the algebra are the equivalence classes under $\approx$. The familiar *bisimulation algorithm* ([19] §3.1; [27] §2) can be reinterpreted algebraically as the construction of a sequence of algebras $\mathcal{S}_k^\mathfrak{M}$ for $k < \omega$, where

$$\mathcal{S}_k^\mathfrak{M} \doteq \{ \|\varphi\|^\mathfrak{M} \mid \varphi \in \mathcal{S}_k(\Phi, \Sigma) \}$$

is the finite Boolean algebra of denotations of *modal* sentences of modal degree $\leq k$. The modal degree measures depth of nesting of modal operators; for example, for hybrid trajectory formulas of the form (3), the degree is $2n + 1$, where n is the length of the discrete trace. It follows that $\mathcal{S}_{k+1}^\mathfrak{M}$ is the smallest Boolean algebra generated by $\mathcal{S}_k^\mathfrak{M} \cup \{ \sigma(a^\mathfrak{M})(A) \mid A \in \mathcal{S}_k^\mathfrak{M} \}$. The algorithm terminates at stage $k + 1$ if $\mathcal{S}_{k+1}^\mathfrak{M} = \mathcal{S}_k^\mathfrak{M}$, in which case the equivalence relation:

$$x \approx_{\mathcal{S}_k^\mathfrak{M}} y \quad \doteq \quad (\forall A \in \mathcal{S}_k^\mathfrak{M}) [x \in A \Leftrightarrow y \in A]$$

is a finite bisimulation equivalence whose equivalence classes are atoms of the algebra $\mathcal{S}_k^\mathfrak{M}$, and $\mathcal{S}_\mu^\mathfrak{M} = \mathcal{S}_k^\mathfrak{M}$.

4 Adding topological and metric tolerance structure

Within modal logic, there is a well-known way of representing a *topology* $\mathcal{T}$ on the state space X of an LTS or Kripke model. From McKinsey and Tarski's work in the 1940's ([31], [32], [36]), the axioms for the box $\Box$ modality of the modal logic **S4** correspond exactly to those of the Kuratowski axioms for the topological interior operator $\text{int}_\mathcal{T}$, and dually, the **S4** diamond $\Diamond$ corresponds to topological closure $\text{cl}_\mathcal{T}$. **S4** is a well-studied modal logic, and is of particular interest in virtue of the 1933 Gödel translation of *Intuitionistic* logic into (classical) **S4**. The relational Kripke semantics for **S4** is in terms of *pre-orders*:

³ The concept is not formally defined here. An analysis of the concept of bisimulation is given in [11]. See also the handbook article [38] §5.3, where it is noted that if one wants to preserve the truth of sentences containing the converse operation, then the notion of bisimulation must be strengthened so as to include respect for the converses of the $a^\mathfrak{M}$.

reflexive and transitive relations $\preceq \subseteq X \times X$, and can be shown to be a special case of the topological semantics via *Alexandroff topologies*, which are in one-one correspondence with pre-orders (see [11]). For background on general topology, see [33], [24].

Let $\mathcal{F}_{\mu, \Box}(\Phi, \Sigma)$ denote the collection of formulas defined as in Definition 3 with an additional clause for a plain $\Box$ modality, with analogous notation for the collection of sentences, and the purely modal fragments. The diamond is defined by the usual negation (de Morgan) duality: $\Diamond \varphi \doteq \neg \Box \neg \varphi$.

Definition 7. *If $\mathfrak{M} = (X, \mathcal{T}, \{a^{\mathfrak{M}}\}_{a \in \Sigma}, \{\|p\|^{\mathfrak{M}}\}_{p \in \Phi})$ is a topologized LTS model then the additional clauses to be added to Definition 4 for the semantics of formulas $\varphi \in \mathcal{F}_{\mu, \Box}(\Phi, \Sigma)$ are:*

$$\|\Box \varphi\|_{\xi}^{\mathfrak{M}} \doteq \text{int}_{\mathcal{T}}(\|\varphi\|_{\xi}^{\mathfrak{M}}) \quad \text{and} \quad \|\Diamond \varphi\|_{\xi}^{\mathfrak{M}} \doteq \text{cl}_{\mathcal{T}}(\|\varphi\|_{\xi}^{\mathfrak{M}})$$

In the enriched language, we can simply express topological properties of *sets* of states. For example, a set $\|\varphi\|^{\mathfrak{M}} \subseteq X$ is, respectively, *open*, *closed*, *dense* or *nowhere dense* (empty interior), with respect to $\mathcal{T}$, exactly when the sentences $\varphi \rightarrow \Box \varphi$, $\Diamond \varphi \rightarrow \varphi$, $\Diamond \varphi$, or $\Diamond \neg \varphi$ are true in $\mathfrak{M}$. The topological boundary of $\|\varphi\|^{\mathfrak{M}}$ is denoted by the sentence $\Diamond \varphi \wedge \neg \Box \varphi$ (and boundary sets are always nowhere dense).

Note that if $X \subseteq \mathbb{R}^n$ is first-order definable in an o-minimal structure $\overline{\mathbb{R}}$, $\mathcal{T}$ is the subspace topology on X inherited from the standard metric topology on $\mathbb{R}^n$ (derived from the order $<$ on $\mathbb{R}$), and $A \subseteq X$ is definable, then $\text{int}_{\mathcal{T}}(A)$ and $\text{cl}_{\mathcal{T}}(A)$ are also definable ([14], Lemma 3.4). Thus if the components of a topologized model $\mathfrak{M}$ are definable in $\overline{\mathbb{R}}$, then the *topological modal algebra*

$$\mathcal{S}_{\Box}^{\mathfrak{M}} \doteq \{\|\varphi\|^{\mathfrak{M}} \mid \varphi \in \mathcal{S}_{\Box}(\Phi, \Sigma)\}$$

of denotations of modal sentences including $\Box$ is also definable. From the perspective of o-minimality, observe that the cells of a *cell decomposition* of a definable $X \subseteq \mathbb{R}^n$ are either open in $\mathbb{R}^n$, or else are boundary sets ([14], Proposition 2.5) – properties expressible in the enriched modal language.

Note that if we want a bisimulation to be truth-preserving with respect to sentences $\varphi \in \mathcal{S}_{\mu, \Box}(\Phi, \Sigma)$, then it must also respect the topology $\mathcal{T}$. For equivalence relations $\approx$, this amounts to the requirement that for each equivalence class B under $\approx$, the closure $\text{cl}_{\mathcal{T}}(B)$ must be a union of equivalence classes, thus either $\text{int}_{\mathcal{T}}(B) = B$ or $\text{int}_{\mathcal{T}}(B) = \emptyset$; in brief, the equivalence classes B are “cell-like”.

OK, so we’ve formally got topologies in the picture, so we should be able to express *some* notion of *continuity*. A sticking point is that the standard notion of continuity is for *functions*, not relations. In purely topological terms, a *function* $f : (X, \mathcal{T}) \rightarrow (Y, \mathcal{S})$ is a *continuous* iff for every open set U in Y , the inverse-image $f^{-1}(U)$ is open in X . The relevant notions for relations $r : (X, \mathcal{T}) \rightsquigarrow (Y, \mathcal{S})$ were introduced by Kuratowski and Bouligand in the 1930’s, and replace the functional inverse-image with the relational $\forall$ - and $\exists$ -pre-image operators.

Definition 8. ([6] §1.4; [1]⁴ Ch. 7; [24] §18.) A relation $r : (X, \mathcal{T}) \rightsquigarrow (Y, \mathcal{S})$ is:

- upper semi-continuous (*u.s.c.*) iff for every open set U in Y , the $\forall$ -pre-image $\tau(r)(U)$ is open in X ;
- lower semi-continuous (*l.s.c.*) iff for every open set U in Y , the $\exists$ -pre-image $\sigma(r)(U)$ is open in X ;
- continuous iff it is both *u.s.c.* and *l.s.c.*

When $r : (X, \mathcal{T}) \rightsquigarrow (Y, \mathcal{S})$ is in fact a (single-valued) function, each of the semi-continuity properties is equivalent to functional continuity, since in that case, the two relational pre-image operators collapse to the familiar inverse-image operator: $\sigma(r) = \tau(r) = r^{-1}$. Logics of continuous functions are developed in [10].

The two semi-continuity properties are simply expressible in the language of the topological μ -calculus by the formulas (sentence schemes):

$$[a]\Box Z \rightarrow \Box[a]Z \quad \text{and} \quad \langle a \rangle \Box Z \rightarrow \Box \langle a \rangle Z \quad (20)$$

In dual form, upper semi-continuity can be read as preservation of *closed* sets by the familiar $\exists$ -pre-image $\sigma(r) = \mathbf{Pre}(r)$:

$$\Diamond \langle a \rangle Z \rightarrow \langle a \rangle \Diamond Z$$

From these simple characterizations of the semi-continuity properties, it follows purely formally that each of the properties is inherited under finite relational compositions and finite relational unions (sums). Inheritance of continuity properties under infinitary fixed-point quantification is a topic of continuing investigation.

So far, the discussion of continuity is still rather formal, and a tad insubstantial. But in the case of *compact metric spaces*, we get to see some meat on the bones.

Proposition 1. ([1] Ch.7, Proposition 11) *For relations $r : X \rightsquigarrow Y$ where X and Y are compact metric spaces and the direct image $r(x) \subseteq Y$ for each $x \in X$ is closed, the following are equivalent:*

1. r is *u.s.c.*;
2. for all $x \in X$ and all $\epsilon > 0$, there is a $\delta > 0$ such that for all $x' \in X$ and $y' \in Y$,

$$d_X(x, x') < \delta \quad \text{and} \quad x' \xrightarrow{r} y' \quad \Rightarrow \quad (\exists y \in Y) [x \xrightarrow{r} y \quad \text{and} \quad d_Y(y, y') < \epsilon]$$

3. as a subset of $X \times Y$, (the graph of) r is closed;

⁴ Note that in [6], [7], Aubin uses the terms “core” and “inverse-image” instead of universal and existential pre-image, while in [1], Akin uses has but has neither names nor notation for the pre-image operators.

4. $\tilde{r} : Y \rightsquigarrow X$ is u.s.c.

The following are also equivalent:

1. r is l.s.c.;
2. for all $x \in X$ and all $\epsilon > 0$, there is a $\delta > 0$ such that for all $x' \in X$ and $y \in Y$,

$$d_X(x, x') < \delta \text{ and } x \xrightarrow{r} y \Rightarrow (\exists y' \in Y)[x' \xrightarrow{r} y' \text{ and } d_Y(y, y') < \epsilon]$$

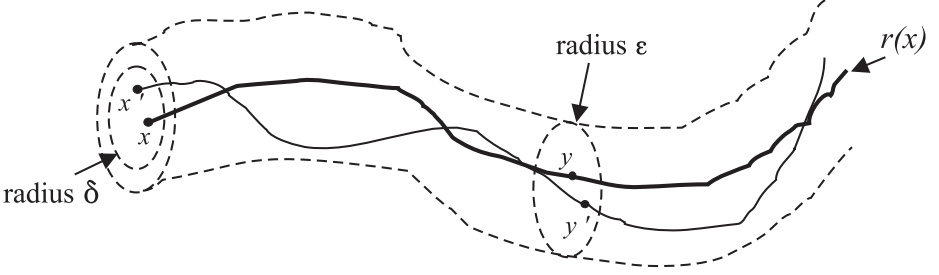


Fig. 4. The u.s.c. property in the compact metric setting.

The metric u.s.c. property says that if an *input* x' is within δ of x , then every point y' in the *output* or image $r(x')$ is contained within an ϵ “ball” or “tube” around $r(x)$. For the orbit relation $f : X \rightsquigarrow X$ of a semi-flow $\phi : X \times \mathbb{R}^+ \rightarrow X$ (defined in (11)), where $f(x) = \{\phi(x, t) \mid t \in \mathbb{R}^+\}$ is the positive trajectory from x , the picture really is that of an ϵ -tube: if $d_X(x, x') < \delta$ then the trajectory $f(x')$ lies inside an ϵ -tube around the trajectory $f(x)$, as illustrated in Figure 4. The idea is certainly reminiscent of the “tube neighborhoods” in the work of Gupta, Henzinger and Jagadeesan [17] on *robust timed automata*; the interest in that paper is on metrics on trajectories $\tau \in (\Phi \times \mathbb{R}^{>0})^*$, where Φ is a finite alphabet of event names.

When X is a compact metric space, $\phi : X \times \mathbb{R}^+ \rightarrow X$ is a continuous semi-flow, and $T \subseteq \mathbb{R}^+$ is compact, the restricted orbit relation $f^T : X \rightsquigarrow X$ given by $f^T(x) = \{\phi(x, t) \mid t \in T\}$ has a closed graph and hence is u.s.c. ([1], Ch. 6). This leads to the following result on continuity properties of *both* sort of transition relations in an LTS model of a hybrid automaton.

Proposition 2. *Let $\mathfrak{M}_{\mathcal{H}}$ be the LTS model of a hybrid automaton, as in Definition 2. Assume that each $X_q \subseteq \mathbb{R}^n$ is compact in the standard topology on $\mathbb{R}^n$. Let $\mathcal{T}_q$ be the subspace topology on X_q , and assume the semi-flow $\phi_q : X_q \times \mathbb{R}^+ \rightarrow X_q$ is continuous.*

1. *If Inv_q is closed in $\mathcal{T}_q$, and time-bounded under ϕ_q , in the sense that there is a $t_q > 0$ such that for all $x \in \text{Inv}_q$ and all $t > t_q$, $\phi_q(x, t) \notin \text{Inv}_q$, then the relation $e_q : X_q \rightsquigarrow X_q$ defined by $e_q = f_q \cap (\text{Inv}_q \times \text{Inv}_q)$ is u.s.c..*

2. If $\text{Grd}_{q,q'} \subseteq X_q$ and $\text{Inv}_{q'} \subseteq X_{q'}$ are both closed, in $\mathcal{T}_q$ and $\mathcal{T}_{q'}$ respectively, and the graph of $r_{q,q'} : X_q \rightsquigarrow X_{q'}$ is closed, then the relation $c_{q,q'} : X_q \rightsquigarrow X_{q'}$ defined by $c_{q,q'} = r_{q,q'} \cap (\text{Grd}_{q,q'} \times \text{Inv}_{q'})$ is u.s.c..

The point is that the u.s.c. property is sufficiently attractive that we may wish it to be the case that *all* our transition relations possess it. From our observations above, all finite compositions and unions of the e_q and $c_{q,q'}$ will be u.s.c. if the e_q and $c_{q,q'}$ are u.s.c.. Note also that for the *constant* jump relations $c_{q,q'} = \text{Grd}_{q,q'} \times \text{Rst}_{q,q'}$ of [27], $c_{q,q'}$ is u.s.c. when both $\text{Grd}_{q,q'}$ and $\text{Rst}_{q,q'}$ are closed.

When the relations $e_q : X_q \rightsquigarrow X_q$ and $c_{q,q'} : X_q \rightsquigarrow X_{q'}$ are lifted to relations $X \rightsquigarrow X$, the issue arises as to what is the appropriate topology on the *hybrid state space* $X \subseteq Q \times \mathbb{R}^n$? Taking the X_q equipped with their standard topology from $\mathbb{R}^n$, the question then becomes: what topology $\mathcal{T}_Q$ on the finite discrete state space Q ? One reasonable choice is that Q really is *discrete* and has no topological structure, which amounts to taking $\mathcal{T}_Q$ to be the discrete topology. Then the lifted relations will be u.s.c. or l.s.c. whenever their unlifted counterparts are. An alternative reasonable choice is to consider Q as structured by the control graph $G \subseteq Q \times Q$, so take $\mathcal{T}_Q = \mathcal{T}_G$ to be the (Alexandroff) topology determined by the reflexive-transitive closure $\preceq_G$ of G . The open (closed) sets in $\mathcal{T}_G$ are those $P \subseteq Q$ that are up- (down-) *invariant* under $\preceq_G$; the clopen sets in $\mathcal{T}_G$ are *cycles* under G . The inherited topology on $X \subseteq Q \times \mathbb{R}^n$, and the continuity properties, are more complicated, and under current investigation.

Metric structure on the state space of an LTS model can be used to define explicit *metric tolerance relations* that allow us to express such properties as *being within ϵ of a set*, for a particular $\epsilon > 0$. Again, the resources of modal logic come into play. For X a metric space and $\epsilon > 0$, define a relation of ϵ -tolerance or ϵ -indiscernability (ϵ) : $X \rightsquigarrow X$ by:

$$x(\epsilon)x' \quad \text{iff} \quad d_X(x, x') < \epsilon \quad (21)$$

Such a relation is *reflexive* and *symmetric*, but not transitive. My source for the notion of a tolerance relation is Smyth's [37]. A motivating idea in that paper, which is traced back to Poincaré's *The Value of Science* (1905) and independently, to the topologist Zeeman in the early 1960's, is that *perceptual* or *physical* continua, as opposed to the *idealized* continua of classical mathematics, are of finite or countable cardinality and are structured by a relation of indiscernability that is reflexive and symmetric, but not transitive. In [1] Ch.1, the relation (ϵ) goes by the name V_ϵ .

Formally, we extend the alphabet Σ of transition labels with a new symbol ϵ . Interpreting the new modalities $\langle \epsilon \rangle$ and $[\epsilon]$ in the standard way by the pre-image operators $\sigma(\epsilon)$ and $\tau(\epsilon)$, the sentence $\langle \epsilon \rangle \varphi$ denotes the ϵ -ball around $\|\varphi\|^{\mathfrak{M}}$, or the ϵ -closure of $\|\varphi\|^{\mathfrak{M}}$ – that is, the set of states within ϵ of *some* point in $\|\varphi\|^{\mathfrak{M}}$, while $[\epsilon]\varphi$ denotes the ϵ -interior of $\|\varphi\|^{\mathfrak{M}}$ – that is, the set of states *all* of whose

ϵ -neighbors are in $\|\varphi\|^{\mathfrak{M}}$. The modalities for symmetric and reflexive relations are axiomatized by the modal logic **KT****B**; see [9] §4.3.

The combination of topological and tolerance structure opens up new possibilities. For example ([1] Ch.1, Corollary 2), if $a^{\mathfrak{M}} : X \rightsquigarrow X$ is u.s.c. in a compact metric space X , then for each *closed* set $\|\varphi\|^{\mathfrak{M}} \subseteq X$, and each $\epsilon > 0$, there is a $\delta > 0$ such that the sentence

$$\langle \delta \rangle \langle a \rangle \varphi \rightarrow \langle a \rangle \langle \epsilon \rangle \varphi \quad (22)$$

is true in $\mathfrak{M}$.

Metric tolerance structure can be used to define “imperfect precision” hybrid trajectories. In the LTS model $\mathfrak{M}_{\mathcal{H}}$ of a hybrid automaton $\mathcal{H}$, suppose that on each projection $X_q \subseteq \mathbb{R}^n$, we have a metric tolerance $(\delta_q) : X_q \rightsquigarrow X_q$ for some given $\delta_q > 0$. Then instead of considering “perfect precision” trajectories formed from the simple alternation of constrained evolution and controlled jump relations, as in (3), we might want to consider transition sequences:

$$e_{q_0} \cdot \delta_{q_0} \cdot c_{q_0, q_1} \cdot e_{q_1} \cdot \delta_{q_1} \cdot c_{q_1, q_2} \cdot e_{q_2} \cdots e_{q_{k-1}} \cdot \delta_{q_{k-1}} \cdot c_{q_{k-1}, q_k} \cdot e_{q_k} \quad (23)$$

Operationally, this can be construed as allowing metric “gaps” of up to size δ_q between the *decision* to make a controlled switch $c_{q, q'}$, and the point at which such a *switch actually occurs*. Defining $(\delta) : X \rightsquigarrow X$ to be the union of each of the lifted relations (δ_q) , the dynamics of the class of all “ δ -imperfect” hybrid trajectories with finite discrete traces are captured by the dual fixed-point modalities

$$\langle \mathbf{h}_{\delta} \rangle \varphi \doteq \mu Z. \langle \mathbf{e} \rangle \varphi \vee \langle \mathbf{e} \rangle \langle \delta \rangle \langle \mathbf{c} \rangle Z \quad \text{and} \quad [\mathbf{h}_{\delta}] \varphi \doteq \nu Z. [\mathbf{e}] \varphi \wedge [\mathbf{e}] \langle \delta \rangle [\mathbf{c}] Z \quad (24)$$

Alternatively, one could “relax” the definition of the constrained evolution relation, and take

$$\langle \acute{\mathbf{e}}_q \rangle Z \leftrightarrow \langle \delta_q \rangle \mathbf{Inv}_q \wedge \langle \mathbf{f}_q \rangle (Z \wedge \mathbf{Inv}_q)$$

that is, $\acute{e}_q = f_q \cap (Inv_q \times \sigma(\delta_q)Inv_q)$, where the revised convexity property is:

$$\langle \check{\mathbf{f}}_q \rangle \mathbf{Inv}_q \wedge \langle \mathbf{f}_q \rangle \langle \delta_q \rangle \mathbf{Inv}_q \rightarrow \langle \delta_q \rangle \mathbf{Inv}_q$$

which says: curves along ϕ_q that start in Inv_q and end in $\sigma(\delta_q)Inv_q$ lie inside $\sigma(\delta_q)Inv_q$.

5 Deductive Proof Systems

We present simple Hilbert-style axiomatic proof systems for the logics of interest. The axiomatizations are not intended to be minimal; rather, they are meant to

serve as a useful reference list. In particular, we give the axioms and rules for both of the dual diamond and box modalities. Kozen's axiomatization $\mathbf{L}_\mu$ [23] forms the foundation, with extensions developed in a modular fashion. So far, we have identified **S4** for topological and relational pre-order modalities, and **KTB** for tolerance relations. A further candidate is **S5**, the modal logic of *equivalence relations*: we can give modal representation to any partition of the state space of our choosing; bisimulation equivalences spring to mind. **S5** is also the base of *logics of knowledge* [16]: the knowledge of an agent is modeled by the equivalence relation of indistinguishability relative to its knowledge base.

Equivalent Gentzen sequent-style proof systems for the μ -calculus are presented in [5], [8], and also in [40].

Definition 9. *The Hilbert-style proof system for the logic $\mathbf{L}_\mu$ has the following axioms: for transition labels $a \in \Sigma$, propositional variables $Z, W \in \text{PVar}$, and formulas $\varphi \in \mathcal{F}_\mu(\Phi, \Sigma)$,*

CP : *axioms of classical propositional logic*

$$\vee\text{-}\langle a \rangle : \langle a \rangle(Z \vee W) \leftrightarrow (\langle a \rangle Z \vee \langle a \rangle W) \qquad \mathbf{ff}\text{-}\langle a \rangle : \langle a \rangle \mathbf{ff} \leftrightarrow \mathbf{ff}$$

$$\wedge\text{-}[a] : [a](Z \wedge W) \leftrightarrow ([a]Z \wedge [a]W) \qquad \mathbf{tt}\text{-}[a] : [a] \mathbf{tt} \leftrightarrow \mathbf{tt}$$

$$\mu\text{-f.p.} : \varphi[Z := \mu Z.\varphi] \rightarrow \mu Z.\varphi \qquad \nu\text{-f.p.} : \nu Z.\varphi \rightarrow \varphi[Z := \nu Z.\varphi]$$

and the inference rules, for formulas $\varphi, \psi \in \mathcal{F}_\mu(\Phi, \Sigma)$:

$$\text{modus ponens:} \quad \frac{\varphi, \varphi \rightarrow \psi}{\psi}$$

$$\text{substitution:} \quad \frac{\varphi}{\varphi[Z := \psi]}$$

$$\langle a \rangle\text{-monotonicity:} \quad \frac{\varphi \rightarrow \psi}{\langle a \rangle \varphi \rightarrow \langle a \rangle \psi}$$

$$[a]\text{-monotonicity:} \quad \frac{\varphi \rightarrow \psi}{[a] \varphi \rightarrow [a] \psi}$$

$$\mu\text{-least-f.p.:} \quad \frac{\varphi[Z := \psi] \rightarrow \psi}{\mu Z.\varphi \rightarrow \psi}$$

$$\nu\text{-greatest-f.p.:} \quad \frac{\psi \rightarrow \varphi[Z := \psi]}{\psi \rightarrow \nu Z.\varphi}$$

$$\text{Hoare composition:} \quad \frac{\psi \rightarrow \langle a \rangle \chi \quad \chi \rightarrow \langle b \rangle \varphi}{\psi \rightarrow \langle a \rangle \langle b \rangle \chi}$$

$$\text{Hoare composition:} \quad \frac{\psi \rightarrow [a] \chi \quad \chi \rightarrow [b] \varphi}{\psi \rightarrow [a][b] \chi}$$

We write: $\mathbf{L}_\mu \vdash \varphi$ for formulas $\varphi \in \mathcal{F}_\mu(\Phi, \Sigma)$ if there is a proof of φ in $\mathbf{L}_\mu$.

The axioms and monotonicity rules for $\langle a \rangle$ and $[a]$ together assert they are *normal* diamond (possibility) and box (necessity) modalities ([9] Ch. 4); they are equivalent to system **K** (for Kripke), the logic of generic binary relations. In the language of [26], $\langle a \rangle$ denotes a *normal* and *finitely additive* operator on a Boolean algebra. The Hoare composition rules follow readily from monotonicity. As always, we assume substitutions $\varphi[Z := \psi]$ are legitimate ones; i.e. no capture of free variables.

The axioms and rules for the fixed-point quantifiers assert what they ought: that $\mu Z.\varphi$ ($\nu Z.\varphi$) is the least (greatest) fixed point of the operator defined by φ .

Each of the rules is readily verified to be *truth-preserving*, in the sense that for any LTS model $\mathfrak{M}$, if the hypotheses of a rule is true in $\mathfrak{M}$ then the conclusion is true in $\mathfrak{M}$. From the verification that the each of the axioms is true in every LTS model, we then get *soundness*: if $\mathbf{L}_\mu \vdash \varphi$ then $\mathfrak{M} \models \varphi$ for all LTS models $\mathfrak{M}$ of signature (Φ, Σ) .

Definition 10. *The Hilbert-style proof system for the logic $\mathbf{L}_\mu + \mathbf{S4}$ in the language $\mathcal{F}_{\mu, \square}(\Phi, \Sigma)$ is obtained from that of $\mathbf{L}_\mu$ by adding the normality axioms and rules for $\diamond$ and $\square$, together with: for propositional variables $Z \in \text{PVar}$,*

$$\begin{array}{ll} \mathbf{T}\diamond : & Z \rightarrow \diamond Z \\ \mathbf{4}\diamond : & \diamond \diamond Z \rightarrow \diamond Z \end{array} \qquad \begin{array}{ll} \mathbf{T}\square : & \square Z \rightarrow Z \\ \mathbf{4}\square : & \square Z \rightarrow \square \square Z \end{array}$$

The proof system for the logic $\mathbf{L}_\mu + \mathbf{S4} + C_a$ is that of $\mathbf{L}_\mu + \mathbf{S4}$ together with C_a , where C_a is one or more of the semi-continuity axiom schemes:

$$\begin{array}{ll} \mathbf{usc}\langle a \rangle : & \diamond \langle a \rangle Z \rightarrow \langle a \rangle \diamond Z \\ \mathbf{lsc}\langle a \rangle : & \langle a \rangle \square Z \rightarrow \square \langle a \rangle Z \end{array} \qquad \begin{array}{ll} \mathbf{usc}[a] : & [a] \square Z \rightarrow \square [a] Z \\ \mathbf{lsc}[a] : & \diamond [a] Z \rightarrow [a] \diamond Z \end{array}$$

In the relational (preorder) semantics for **S4**, the **T** axioms correspond to reflexivity, while the **4** axioms correspond to transitivity. Extensions of the Hoare composition rules:

$$\frac{\psi \rightarrow [a] \square \chi \quad \chi \rightarrow [b] \square \varphi}{\psi \rightarrow [a] [b] \square \varphi} \quad \text{and} \quad \frac{\psi \rightarrow \langle a \rangle \square \chi \quad \chi \rightarrow \langle b \rangle \square \varphi}{\psi \rightarrow \langle a \rangle \langle b \rangle \square \varphi}$$

can be derived in the systems $\mathbf{L}_\mu + \mathbf{S4} + \mathbf{usc}[a] + \mathbf{usc}[b]$ and $\mathbf{L}_\mu + \mathbf{S4} + \mathbf{lsc}\langle a \rangle + \mathbf{lsc}\langle b \rangle$ respectively.

Definition 11. *The Hilbert-style proof system for the logic $\mathbf{L}_\mu + \mathbf{KTB}$ in the language $\mathcal{F}_\mu(\Phi, \Sigma \cup \{\epsilon\})$ is obtained from that of $\mathbf{L}_\mu$ by adding the normality axioms and rules for $\langle \epsilon \rangle$ and $[\epsilon]$; the axioms $\mathbf{T}\langle \epsilon \rangle$ and $\mathbf{T}[\epsilon]$; and also:*

$$\mathbf{B}\langle \epsilon \rangle : \quad \langle \epsilon \rangle [\epsilon] Z \rightarrow Z \qquad \mathbf{B}[\epsilon] : \quad Z \rightarrow [\epsilon] \langle \epsilon \rangle Z$$

The **B** axioms express that tolerance relations (ϵ) are symmetric.

Definition 12. *The Hilbert-style proof system for the logic $\mathbf{L}_\mu + \mathbf{S5}$ in the language $\mathcal{F}_\mu(\Phi, \Sigma \cup \{\approx\})$ is obtained from that of $\mathbf{L}_\mu$ by adding the normality axioms and rules for $\langle \approx \rangle$ and $[\approx]$; the axioms $\mathbf{T}\langle \approx \rangle$, $\mathbf{T}[\approx]$, $\mathbf{4}\langle \approx \rangle$ and $\mathbf{4}[\approx]$; and also:*

$$\mathbf{5}\langle \approx \rangle : \langle \approx \rangle[\approx]Z \rightarrow [\approx]Z \quad \mathbf{5}[\approx] : \langle \approx \rangle Z \rightarrow [\approx]\langle \approx \rangle Z$$

The $\mathbf{5}$ axioms express that $\approx$ is a *Euclidean* relation: if $x \approx y$ and $x \approx z$ then $y \approx z$. And reflexive, transitive and Euclidean binary relations are exactly equivalence relations. Under the knowledge interpretation of $\mathbf{S5}$, the axiom $\mathbf{5}[\approx]$ is usually referred to as the axiom of *negative introspection*: $\neg[\approx]\varphi \rightarrow [\approx]\neg[\approx]\varphi$, which reads: “if it is not the case that agent A knows φ , then agent A knows that it is not the case that she knows φ ”.

Walukiewicz has recently established the completeness of the Kozen axiomatization with respect to the standard set-theoretic semantics for the μ -calculus.

Theorem 1. ([39],[40]) Soundness and Completeness of $\mathbf{L}_\mu$ (set-theoretic semantics)

For all formulas $\varphi \in \mathcal{F}_\mu(\Phi, \Sigma)$,

$\mathbf{L}_\mu \vdash \varphi$ iff $\mathfrak{M} \models \varphi$ for all LTS models $\mathfrak{M}$ of signature (Φ, Σ) .

The completeness part of the cited theorem is stated in the form: if φ is *unsatisfiable* in every LTS model $\mathfrak{M}$, i.e. $\|\varphi\|_\xi^{\mathfrak{M}} = \emptyset$ for all assignments ξ in $\mathcal{P}(X)$, then $\neg\varphi$ is provable in $\mathbf{L}_\mu$. Walukiewicz’s proof is very intricate, proceeding by first contracting to a subclass of “nice” formulas, and then producing a “tableaux refutation” of unsatisfiable formulas of nice form, where such a refutation in turn implies that the negation of the given formula is provable in $\mathbf{L}_\mu$. Topics of continuing enquiry include whether the Walukiewicz proof can be extended to cover specific modal enrichments of $\mathbf{L}_\mu$, and the relationship between his tableaux refutation system and a tableaux proof system for the μ -calculus and polymodal extensions, in the style of [35] and [10].

The algebraic semantics of Kwiatkowska *et al.* [5], [8], provide a framework for extending Stone duality theory to the algebra of fixed-points. Their proof of completeness for modal μ -frames starts with the Lindenbaum algebra $\mathcal{F}_{\mathbf{L}_\mu}$ of formulas in $\mathcal{F}_\mu(\Phi, \Sigma)$ modulo provable equivalence in $\mathbf{L}_\mu$, then realizes the abstract μ -algebra as a canonical LTS model $\mathfrak{M}_{\mathbf{L}_\mu}$ with state space the Stone space $X = \text{Ult}(\mathcal{F}_{\mathbf{L}_\mu})$ of (Boolean) ultrafilters in $\mathcal{F}_{\mathbf{L}_\mu}$, together with the canonical μ -algebra $\mathcal{A}_{\mathbf{L}_\mu} = \text{Clop}(\text{Ult}(\mathcal{F}_{\mathbf{L}_\mu})) \cong \mathcal{F}_{\mathbf{L}_\mu}$ of subsets of X clopen in the Stone topology. For each $a \in \Sigma$, and $\mathfrak{M} = \mathfrak{M}_{\mathbf{L}_\mu}$, the relations $a^{\mathfrak{M}}$ on X are defined by: $x \xrightarrow{a^{\mathfrak{M}}} y$ iff $(\forall \bar{\varphi} \in \mathcal{F}_{\mathbf{L}_\mu}) [\overline{[a]\varphi} \in x \Rightarrow \bar{\varphi} \in y]$. The formal statement of the result is as follows.

Theorem 2. ([5]) Soundness and Completeness of $\mathbf{L}_\mu$ (algebraic semantics)

For all formulas $\varphi \in \mathcal{F}_\mu(\Phi, \Sigma)$,

$\mathbf{L}_\mu \vdash \varphi$ iff $(\mathfrak{M}, \mathcal{A}) \models \varphi$ for all modal μ -frames $(\mathfrak{M}, \mathcal{A})$ of signature (Φ, Σ) .

In [8] §6, it is established if $(\mathfrak{M}, \mathcal{A})$ is a *descriptive* modal μ -frame, then $(\mathfrak{M}, \mathcal{A})$ is in semantic agreement with $\mathfrak{M}$. In particular, the canonical frame $(\mathfrak{M}_{\mathbf{L}_\mu}, \mathcal{A}_{\mathbf{L}_\mu})$ is descriptive, and thus in semantic agreement with the underlying LTS model $\mathfrak{M}_{\mathbf{L}_\mu}$. Thus the “easy” algebraic proof of completeness can be used to give an alternative proof of completeness of $\mathbf{L}_\mu$ with respect to the standard set-theoretic semantics, as stated in Theorem 1.

The Kwiatkowska algebraic completeness proof extends quite smoothly to normal polymodal extensions of the μ -calculus, including topological **S4** extensions with semi-continuity axioms. For example, if $\mathbf{L} = \mathbf{L}_\mu + \mathbf{S4} + \{\mathbf{usc}[a] + \mathbf{lsc}\langle a \rangle\}_{a \in \Sigma}$, the topology on the canonical model $\mathfrak{M}_{\mathbf{L}}$ comes from a relation $\preccurlyeq$ on $X = \text{Ult}(\mathcal{F}_{\mathbf{L}})$ defined in the same way as the relations $a^{\mathfrak{M}_{\mathbf{L}}}$ as above. The **S4** axioms ensure that the relation $\preccurlyeq$ is a preorder, so the topology is Alexandroff, and from the semi-continuity axiom schemes, one proves that each of the relations $a^{\mathfrak{M}_{\mathbf{L}}}$ have the corresponding semi-continuity property. A more detailed treatment is given in [12].

6 Discussion

We have developed a family of expressively rich and usable logical systems and broadened horizons for the formal analysis of hybrid dynamical systems. In addition to those mentioned in the text, further lines of enquiry include the following.

- Investigation of non-deterministic continuous dynamics, in the form of set-valued or parametrized semi-flows, and their topological properties. Our relation-based view of dynamics is of course conducive to such generalizations.
- A deeper investigation of relations (definable families) in o-minimal structures, and of the use of finite cell-decomposition in the construction of *topological* bisimulations.
- Further investigation of finite sub-topologies of the standard topology on $X \subseteq \mathbb{R}^n$, and semi-continuity properties of relations in such topologies, pursuing themes developed in [11].
- Application to hybrid systems of the theory of knowledge in multi-agent settings and its formalization in **S5** based logics of knowledge.
- LTS models and μ -calculus specifications of *hybrid petri nets*. One approach is to take the state space X to be a set of finite partial functions $x : P \rightsquigarrow \mathbb{R}$ (equivalently, variable-length vectors over $\mathbb{R}$), where P is the finite set of *places* of the net.
- Application of game-theoretic methods for the μ -calculus, and related work on automata over transition systems; e.g. [25], [22].
- Investigation of *tableaux proof systems* for polymodal logics and the μ -calculus, in the style of [35] and [10].
- Investigation of *Intuitionistic* (constructive) logics for hybrid systems, using topological semantics and **S4** as a bridge between the classical and constructive worlds.

Acknowledgments: I would like to thank Prof. Anil Nerode, David Cook, Joe Miller, Suman Ganguli and Prof. Dexter Kozen for valuable conversations, and Xi Krump for graphic artistry.

References

1. E. Akin, *The General Topology of Dynamical Systems*, Graduate Studies in Mathematics 1 (American Mathematical Society, Providence, 1993).
2. R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis and S. Yovine, "The Algorithmic Analysis of Hybrid Systems", *Theoretical Computer Science* **138** (1995) 3-34.
3. R. Alur and D. L. Dill, "A Theory of Timed Automata", *Theoretical Computer Science* **126** (1994) 183-235.
4. R. Alur, T. A. Henzinger and P.-H. Ho, "Automatic Symbolic Verification of Embedded Systems", *IEEE Transactions on Software Engineering* **22** (1996) 181-201.
5. S. Ambler, M. Kwiatkowska and N. Measor, "Duality and the Completeness of the Modal μ -calculus", *Theoretical Computer Science* **151** (1995) 3-27.
6. J.-P. Aubin and H. Frankowska, *Set-Valued Analysis* (Birkhäuser, Boston, 1990).
7. J.-P. Aubin, *Viability Theory* (Birkhäuser, Boston, 1991).
8. M. M. Bonsangue and M. Z. Kwiatkowska, "Reinterpreting the Modal μ -calculus", in A. Ponse, M. de Rijke and Y. Venema (eds.), *Modal Logic and Process Algebra*, CSLI Lecture Notes **53** (CSLI Publications, Stanford, 1995); 65-83.
9. B. F. Chellas, *Modal Logic: An Introduction* (Cambridge University Press, 1980).
10. J. M. Davoren, *Modal Logics for Continuous Dynamics*, PhD dissertation, Department of Mathematics, Cornell University, January 1998.
11. J. M. Davoren, "Topologies, Continuity and Bisimulations", presented at Fixed Points in Computer Science, Brno, August 1998; submitted for publication.
12. J. M. Davoren, "On Continuous Dynamics and Modal Logics", in preparation.
13. C. Daws, A. Olivero, S. Tripakis and S. Yovine, "The Tool KRONOS", in R. Alur, T. A. Henzinger and E. D. Sontag (eds.), *Hybrid Systems III*, Lecture Notes in Computer Science **1066** (Springer-Verlag, Berlin, 1996); 208-219.
14. L. van den Dries, *Tame Topology and O-minimal Structures*, London Mathematical Society Lecture Notes Series **248** (Cambridge University Press, 1998).
15. E. A. Emerson, "Modal Checking and the Mu-calculus", in N. Immerman and P. G. Kolaitis (eds.), *Descriptive Complexity and Finite Models*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science **31** (American Mathematical Society, Providence, 1997); 185-208.
16. R. Fagin, J. Y. Halpern, Y. Moses and M. Y. Vardi, *Reasoning About Knowledge* (MIT Press, Cambridge MA, 1995).
17. V. Gupta, T. A. Henzinger and R. Jagadeesan, "Robust Timed Automata", in O. Maler (ed.), *Hybrid and Real-Time Systems: International Workshop HART'97, March 1997*, Lecture Notes in Computer Science **1201** (Springer-Verlag, Berlin, 1997); 331-345.
18. L. Henkin, "Completeness in the Theory of Types", *Journal of Symbolic Logic* **15** (1950) 81-91.
19. T. A. Henzinger, "The Theory of Hybrid Automata", *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science (LICS '96)* (IEEE Computer Society Press, 1996); 278-292.

20. T. A. Henzinger, O. Kupferman, and S. Qadeer, "From *Pre-historic* to *Post-modern* Symbolic Model Checking", *Proceedings of the Tenth International Conference on Computer-aided Verification (CAV 1998)*, Lecture Notes in Computer Science (Springer-Verlag, Berlin, 1998).
21. T. A. Henzinger, Z. Manna and A. Pnueli, "Towards Refining Temporal Specifications into Hybrid Systems", in R. Grossman *et al.* (eds.), *Hybrid Systems*, Lecture Notes in Computer Science **736** (Springer-Verlag, Berlin, 1993), 60-76.
22. M. J. Hollenberg, *Logic and Bisimulation*, PhD dissertation, Department of Philosophy, Utrecht University, March 1998.
23. D. Kozen, "Results on the Propositional μ -Calculus", *Theoretical Computer Science* **27** (1983) 333-354.
24. K. Kuratowski, *Topology, Volume 1*, revised edition (Academic Press, New York, 1966). Translated by J. Jaworowski from the French 1958 edition of *Topologie, Volume 1*, Polska Akademia Nauk Monografie Matematyczne, Tom **21** (Państwowe Wydawnictwo Naukowe, Warsaw, 1958).
25. D. Janin and I. Walukiewicz, "On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic", *Proceedings of the Seventh International Conference on Concurrency Theory (CONCUR'96)*, Lecture Notes in Computer Science **1119** (Springer-Verlag, Berlin, 1996), 263-277.
26. B. Jónsson and A. Tarski, "Boolean Algebras with Operators, Part I", *American Journal of Mathematics* **73** (1951) 891-939.
27. G. Lafferriere, G. J. Pappas and S. Sastry, "O-Minimal Hybrid Systems", Technical report UCB/ERL M98/29, Department of Electrical Engineering and Computer Science, University of California at Berkeley, May 1998.
28. G. Lafferriere, G. J. Pappas and S. Yovine, "Decidable Hybrid Systems", Technical report UCB/ERL M98/39, Department of Electrical Engineering and Computer Science, University of California at Berkeley, June 1998.
29. Z. Manna and A. Pnueli, "Verifying Hybrid Systems", in R. Grossman *et al.* (eds.), *Hybrid Systems*, Lecture Notes in Computer Science **736** (Springer-Verlag, Berlin, 1993), 4-35.
30. Z. Manna and H. B. Sipma, "Deductive Verification of Hybrid Systems Using STeP", in T. A. Henzinger and S. Sastry (eds), *Hybrid Systems-Computation and Control: Proceedings of the First International Workshop (HSCC '98)*, Lecture Notes in Computer Science **1386**, (Springer-Verlag, Berlin, 1998).
31. J. C. C. McKinsey, "A Solution of the Decision Problem for the Lewis Systems S2 and S4, with an Application to Topology", *Journal of Symbolic Logic* **6** (1941) 117-134.
32. J. C. C. McKinsey and A. Tarski, "The Algebra of Topology", *Annals of Mathematics* **45** (1944) 141-191.
33. J. R. Munkres, *Topology: A First Course* (Prentice Hall, Englewood Cliffs, 1975).
34. A. Nerode and W. Kohn, "Models for Hybrid Systems: Automata, Topologies, Controllability, Observability", in R. Grossman *et al.* (eds.), *Hybrid Systems*, Lecture Notes in Computer Science **736** (Springer-Verlag, Berlin, 1993), 297-316.
35. A. Nerode and R. Shore, *Logic for Applications* (2nd ed.), Graduate Texts in Computer Science (Springer-Verlag, Berlin, 1997).
36. H. Rasiowa and R. Sikorski, *The Mathematics of Metamathematics*, Polska Akademia Nauk Monografie Matematyczne, Tom **41** (Państwowe Wydawnictwo Naukowe, Warsaw, 1963).
37. M. B. Smyth, "Semi-metrics, closure spaces and digital topology", *Theoretical Computer Science* **151** (1995) 257-276.

38. C. Stirling, "Modal and Temporal Logics", in S. Abramsky, D. M. Gabbay and T. Maibaum (eds.), *Handbook of Logic in Computer Science, Vol. 2: Computational Structures* (Oxford University Press, Clarendon Press, Oxford, 1992), 477-563
39. I. Walukiewicz, "On Completeness of the μ -calculus", *Proceedings of the 8th Annual IEEE Symposium on Logic in Computer Science (LICS '93)* (IEEE Computer Society Press, 1993); 136-146.
40. I. Walukiewicz, "A Note on the Completeness of Kozen's Axiomatization of the Propositional μ -calculus", *Bulletin of Symbolic Logic* **2** (1996) 349-366.

The Extended Linear Complementarity Problem and the Modeling and Analysis of Hybrid Systems

Bart De Schutter* and Bart De Moor**

ESAT-SISTA, K.U.Leuven
Kardinaal Mercierlaan 94
B-3001 Leuven, Belgium

tel: +32-16-321799, fax: +32-16-321970

`bart.deschutter@esat.kuleuven.ac.be`, `bart.demoor@esat.kuleuven.ac.be`

Abstract. First we give a short description of the Extended Linear Complementarity Problem (ELCP), which is a mathematical programming problem. We briefly discuss how this problem can be used in the analysis of discrete event systems and continuous variable systems. Next we show that the ELCP can also be used to model and to analyze hybrid systems. More specifically, we consider a traffic-light-controlled intersection, which can be considered as a hybrid system. We construct a model that describes the evolution of the queue lengths in the various lanes (as continuous variables) as a function of time and we show that this leads to an ELCP. Furthermore, it can be shown that some problems in the analysis of another class of hybrid systems, the “complementary-slackness systems”, also lead to an ELCP.

1 Introduction

The main purpose of this paper is to show that the Extended Linear Complementarity Problem (ELCP) — which is a kind of mathematical programming problem — can be used to model and to analyze certain classes of hybrid systems. The formulation of the ELCP arose from our research on discrete event systems. Furthermore, the ELCP can also be used to analyze some classes of continuous variable systems (i.e., systems that can be modeled using difference or differential equations). Since hybrid systems can be considered as a merge of discrete event systems and continuous variable systems, this leads to the question as to whether the ELCP can also be used in the analysis of hybrid systems. We show that this is indeed the case. More specifically, we consider a traffic-light-controlled intersection — which can be considered as a simple hybrid system. We show that the evolution of the queue lengths at a traffic-light-controlled intersection can be described by an ELCP. Furthermore, the ELCP can also be used to

* Senior research assistant with the F.W.O. (Fund for Scientific Research-Flanders).

** Research associate with the F.W.O.

model another class of hybrid systems, the so-called “complementary-slackness systems”.

This paper is organized as follows. In Section 2 we introduce the Extended Linear Complementarity Problem. In Section 3 we briefly discuss how the ELCP can be used to model and to analyze certain classes of discrete event systems, continuous variable systems and hybrid systems. In Section 4 we consider a traffic-light-controlled intersection and we show how the evolution of the queue lengths in this system can be described by an ELCP. Finally, we present some conclusions and directions for future research in Section 5.

2 The Extended Linear Complementarity Problem

The Extended Linear Complementarity Problem (ELCP) is an extension of the Linear Complementarity Problem, which is one of the fundamental problems in mathematical programming [3]. The ELCP is defined as follows:

Given $A \in \mathbb{R}^{p \times n}$, $B \in \mathbb{R}^{q \times n}$, $c \in \mathbb{R}^p$, $d \in \mathbb{R}^q$ and m subsets $\phi_1, \phi_2, \dots, \phi_m$ of $\{1, 2, \dots, p\}$, find $x \in \mathbb{R}^n$ such that

$$\sum_{j=1}^m \prod_{i \in \phi_j} (Ax - c)_i = 0 \quad (1)$$

subject to $Ax \geq c$ and $Bx = d$, or show that no such x exists.

Equation (1) represents the *complementarity condition* of the ELCP. One possible interpretation of this condition is the following: since $Ax \geq c$, (1) is equivalent to

$$\forall j \in \{1, 2, \dots, m\} : \prod_{i \in \phi_j} (Ax - c)_i = 0 .$$

So we could say that each set ϕ_j corresponds to a group of inequalities of $Ax \geq c$ and that in each group at least one inequality should hold with equality, i.e., the corresponding residue should be equal to 0:

$$\forall j \in \{1, 2, \dots, m\} : \exists i \in \phi_j \text{ such that } (Ax - c)_i = 0 .$$

In general, the solution set of the ELCP defined above consists of the union of faces of the polyhedron defined by the system of linear equations and inequalities ($Ax \geq c$ and $Bx = d$) of the ELCP. In [6,7] we have developed an algorithm to compute the complete solution set of an ELCP. This algorithm yields a description of the solution set of an ELCP by vertices, extreme rays and a basis of the linear subspace corresponding to the largest affine subspace of the solution set. In that way it provides a geometrical insight in the entire solution set of the ELCP and related problems.

We shall now give a brief description of the ELCP algorithm of [6,7]. The algorithm consists of two parts:

- First we determine the set of (finite) vertices $\mathcal{X}^{\text{fin}}$, the set of extreme rays $\mathcal{X}^{\text{ext}}$ and a basis $\mathcal{X}^{\text{cen}}$ of the linear subspace corresponding to the largest affine subspace of the solution set of the ELCP. This is done by iteratively solving the system $Ax \geq c$, $Bx = d$, whereby in the k th step ($k = 1, 2, \dots, p+q$) we compute the intersection of the current solution set with the half-space or hyperplane determined by the k th inequality or equality. We also remove solutions that do not satisfy the complementarity condition.
- Next we determine the set Λ of maximal *cross-complementary* pairs of subsets of $\mathcal{X}^{\text{ext}}$ and $\mathcal{X}^{\text{fin}}$. A pair $(\mathcal{X}_s^{\text{ext}}, \mathcal{X}_s^{\text{fin}})$ is cross-complementary if the sum of any nonnegative combination of the elements of $\mathcal{X}_s^{\text{ext}}$ and any convex combination of the elements of $\mathcal{X}_s^{\text{fin}}$ satisfies the complementarity condition. The set Λ is determined using a kind of backtracking algorithm: we start with a pair of the form $(\emptyset, \{x_k^{\text{f}}\})$ with $x_k^{\text{f}} \in \mathcal{X}^{\text{fin}}$ and then we keep on adding new elements of $\mathcal{X}^{\text{ext}}$ and $\mathcal{X}^{\text{fin}}$ to the current pair in a systematic way until we obtain a pair that is not cross-complementary any more¹. In that case we do a backtracking step. This continues until we have obtained all maximal cross-complementary pairs.

Now any solution x of the ELCP can be written as

$$x = \sum_{x_k^{\text{c}} \in \mathcal{X}^{\text{cen}}} \lambda_k x_k^{\text{c}} + \sum_{x_k^{\text{e}} \in \mathcal{X}_s^{\text{ext}}} \kappa_k x_k^{\text{e}} + \sum_{x_k^{\text{f}} \in \mathcal{X}_s^{\text{fin}}} \mu_k x_k^{\text{f}} \quad (2)$$

for some pair $(\mathcal{X}_s^{\text{ext}}, \mathcal{X}_s^{\text{fin}}) \in \Lambda$ with $\lambda_k \in \mathbb{R}$, $\kappa_k \geq 0$, $\mu_k \geq 0$ and $\sum_k \mu_k = 1$.

For more information on the ELCP algorithm and for a worked example the interested reader is referred to [6].

In [6,7] we have also shown that the general ELCP with rational data is an NP-hard problem.

3 The ELCP and discrete event systems and continuous variable systems

In this section we briefly discuss how the ELCP can be used in the analysis of certain classes of discrete event systems (such as max-linear discrete event systems) and of certain classes of continuous variable systems (such as, e.g., piecewise-linear resistive electrical circuits).

3.1 The ELCP and max-linear time-invariant discrete event systems

Typical examples of discrete event systems (DESSs) are flexible manufacturing systems, subway traffic networks, parallel processing systems, telecommunication

¹ It can be shown that it is sufficient to test only one combination of the elements of $\mathcal{X}_s^{\text{ext}}$ and $\mathcal{X}_s^{\text{fin}}$ to determine whether the pair $(\mathcal{X}_s^{\text{ext}}, \mathcal{X}_s^{\text{fin}})$ is cross-complementary or not.

networks and logistic systems. The class of the DESs essentially contains man-made systems that consist of a finite number of resources (e.g., machines, communications channels, or processors) that are shared by several users (e.g., product types, information packets, or jobs) all of which contribute to the achievement of some common goal (e.g., the assembly of products, the end-to-end transmission of a set of information packets, or a parallel computation).

One of the most characteristic features of a DES is that its dynamics are event-driven as opposed to time-driven: the behavior of a DES is governed by events rather than by ticks of a clock. An event corresponds to the start or the end of an activity. If we consider a production system then possible events are: the completion of a part on a machine, a machine breakdown, or a buffer becoming empty.

In general, the description of the behavior of a DES leads to a model that is nonlinear in conventional algebra. However, there exists a class of DESs for which the model is “linear” when we express it in the max-plus algebra [1,2,4], which has maximization and addition as basic operations. DESs that can be described by such a “linear” model are called *max-linear DESs*. Loosely speaking we could say that the class of max-linear DESs corresponds to the class of deterministic time-invariant DESs in which only synchronization and no concurrency occurs.

The basic operations of the max-plus algebra are maximization (represented by $\oplus$) and addition (represented by $\otimes$). There exists a remarkable analogy between the basic operations of the max-plus algebra on the one hand, and the basic operations of conventional algebra (addition and multiplication) on the other hand. As a consequence many concepts and properties of conventional algebra (such as Cramer’s rule, eigenvectors and eigenvalues, the Cayley-Hamilton theorem, ...) also have a max-plus-algebraic analogue (see, e.g., [1]). Furthermore, this analogy also allows us to translate many concepts, properties and techniques from conventional linear system theory to system theory for max-linear time-invariant DESs. However, there are also some major differences that prevent a straightforward translation of properties, concepts and algorithms from conventional linear algebra and linear system theory to max-plus algebra and max-plus-algebraic system theory for DESs.

If we write down a model for a max-linear DES and if we use the symbols $\oplus$ and $\otimes$ to denote maximization and addition² we obtain a description of the following form:

$$x(k+1) = A \otimes x(k) \oplus B \otimes u(k) \quad (3)$$

$$y(k) = C \otimes x(k) \quad , \quad (4)$$

where x is the state vector, u the input vector and y the output vector. For a manufacturing system $u(k)$ would typically represent the time instants at which

² For matrices A and B these operations are defined by $(A \oplus B)_{ij} = a_{ij} \oplus b_{ij}$ and

$(A \otimes B)_{ij} = \bigoplus_k a_{ik} \otimes b_{kj}$. Note that these definitions closely resemble the definitions

of matrix sum and matrix product of conventional algebra but with $+$ replaced by $\oplus$ and $\times$ replaced by $\otimes$.

raw material is fed to the system for the $(k - 1)$ st time; $x(k)$ the time instants at which the machines start processing the k th batch of intermediate products; and $y(k)$ the time instants at which the k th batch of finished products leaves the system. In analogy with the state space model for linear time-invariant discrete-time systems, a model of the form (3)–(4) is called a *max-linear time-invariant state space model*.

Let $x, r \in \mathbb{R}$. The r th max-plus-algebraic power of x is denoted by $x^{\boxplus r}$ and corresponds to rx in conventional algebra.

Now consider the following problem:

Given $p_1 + p_2$ positive integers $m_1, \dots, m_{p_1+p_2}$ and real numbers a_{ki} , b_k and c_{kij} for $k = 1, \dots, p_1 + p_2$, $i = 1, \dots, m_k$ and $j = 1, \dots, n$, find $x \in \mathbb{R}^n$ such that

$$\bigoplus_{i=1}^{m_k} a_{ki} \otimes \bigotimes_{j=1}^n x_j^{\boxplus c_{kij}} = b_k \quad \text{for } k = 1, \dots, p_1, \quad (5)$$

$$\bigoplus_{i=1}^{m_k} a_{ki} \otimes \bigotimes_{j=1}^n x_j^{\boxplus c_{kij}} \leq b_k \quad \text{for } k = p_1 + 1, \dots, p_1 + p_2. \quad (6)$$

We call (5)–(6) a *system of multivariate max-plus-algebraic polynomial equalities and inequalities*. Note that the exponents may be negative or real.

In [6,10] we have shown that the problem of solving a system of multivariate max-plus-algebraic polynomial equalities and inequalities can be recast as an ELCP. This enables us to solve many important problems that arise in the max-plus algebra and in the system theory for max-linear DESs such as: computing max-plus-algebraic matrix factorizations, performing max-plus-algebraic state space transformations, computing state space realizations of the impulse response of a max-linear time-invariant DES, constructing matrices with a given max-plus-algebraic characteristic polynomial, computing max-plus-algebraic singular value decompositions, computing max-plus-algebraic QR decompositions, and so on [6,7,8,9,10].

Although the analogues of these problems in conventional algebra and linear system theory are easy to solve, the max-plus-algebraic problems are not that easy to solve and for almost all of them the ELCP approach is at present the only way to solve the problem.

For more information on the max-plus-algebra and on max-plus-algebraic system theory for discrete event systems the interested reader is referred to [1,2,4,6,15] and the references given therein.

3.2 The ELCP and piecewise-linear resistive electrical circuits

In this section we consider electrical circuits that may contain the following elements: linear resistive elements, piecewise-linear (PWL) resistors (the resistors are not required to be either voltage or current controlled), and PWL controlled sources (all four types) with one controlling variable (the characteristics may

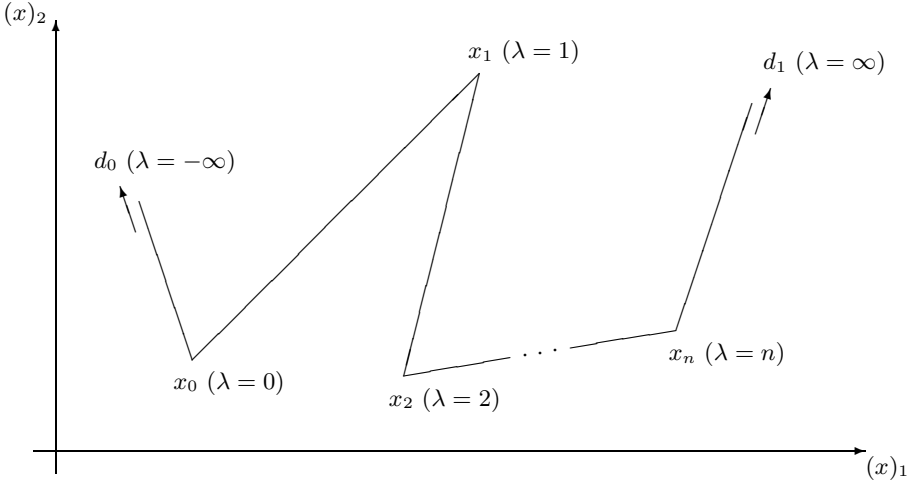


Fig. 1. A one-dimensional PWL curve in $\mathbb{R}^2$ characterized by $n + 1$ breakpoints $x_0, \dots, x_n$ and two directions d_0 and d_1 . The points on this curve can be parameterized by (7) where λ is a real continuous parameter.

be multi-valued). These electrical circuits can be considered as examples of continuous variable systems (i.e., systems that can be modeled using difference or differential equations). In this section we shall show that by using an intelligent parameterization of the PWL characteristics the equations that describe the relations between the voltages and currents in these electrical circuits can be reformulated as (a special case of) an ELCP. For sake of simplicity we consider only two-terminal resistors since they can be described by a one-dimensional PWL manifold³.

If x is a vector, then we define $x^+ = \max(x, 0)$ and $x^- = \max(-x, 0)$, where the operations are performed componentwise. An equivalent definition is:

$$x = x^+ - x^-, \quad x^+, x^- \geq 0, \quad (x^+)^T x^- = 0.$$

It is easy to verify that a one-dimensional PWL curve in $\mathbb{R}^2$ characterized by $n + 1$ breakpoints $x_0, \dots, x_n$ and two directions d_0 and d_1 (see Figure 1) can be parameterized as follows [5,23]:

$$x = x_0 + d_0 \lambda^- + (x_1 - x_0) \lambda^+ + \sum_{k=2}^n (x_k - 2x_{k-1} + x_{k-2}) (\lambda - k + 1)^+ + (d_1 - x_n + x_{n-1}) (\lambda - n)^+, \quad (7)$$

³ If we allow multi-terminal nonlinear resistors, which can be modeled by higher-dimensional PWL manifolds, we shall also obtain an ELCP (See [5]).

with $\lambda \in \mathbb{R}$. Introducing auxiliary variables $\lambda_i = \lambda - i$ yields a description of the following form:

$$\begin{aligned} x &= x_0 + Ay^- + By^+ \\ C(y^+ - y^-) &= d \\ y^+, y^- &\geq 0 \\ (y^+)^T y^- &= 0 \end{aligned}$$

where

$$\begin{aligned} y^- &= [\lambda^- \ \lambda_1^- \ \dots \ \lambda_n^-]^T \\ y^+ &= [\lambda^+ \ \lambda_1^+ \ \dots \ \lambda_n^+]^T \\ A^- &= [d_0 \ 0 \ \dots \ 0] \\ A^+ &= [0 \ x_1 - x_0 \ x_2 - 2x_1 + x_0 \ \dots \ d_1 - x_n + x_{n-1}] \\ B &= \begin{bmatrix} 1 & -1 & 0 & \dots & 0 \\ 1 & 0 & -1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & 0 & \dots & -1 \end{bmatrix} \\ c &= [1 \ 2 \ \dots \ n]^T . \end{aligned}$$

If we extract all nonlinear resistors out of the electrical circuit, the resulting N -port contains only linear resistive elements and independent sources. As a consequence, the relation between the branch currents and voltages of this N -port is described by a system of linear equations. If we combine these equations with the PWL descriptions (7) of the nonlinear resistors, we finally get a system of the form:

$$Mw^+ + Nw^- = q, \quad w^+, w^- \geq 0, \quad (w^+)^T (w^-) = 0, \quad (8)$$

where the vector w contains the parameters λ and λ_i of the PWL descriptions of all the nonlinear resistors. It is easy to verify that (8) can be considered as (a special case of) an ELCP. If we solve (8), we get the complete set of operating points of the electrical circuit.

In a similar way we can determine the driving-point characteristic (i.e., the relation between the input current and the input voltage) and the transfer characteristics of the electrical circuit [23].

In general, the behavior of an electrical network consisting of linear resistors, capacitors, inductors, transformers, gyrators and ideal diodes can be described by a model of the form

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned}$$

subject to the conditions

$$y(t) \geq 0, \quad u(t) \geq 0, \quad (y(t))^T u(t) = 0 \quad (9)$$

(see, e.g., [19]). In order to compute the *stationary points* of such an electrical circuit, we add the condition $\dot{x}(t) = 0$, which leads to a Linear Complementarity Problem [19]. If we replace (9) by more general conditions of the form $w_i \geq 0$, $z_i \geq 0$, $w_i z_i = 0$, where w_i and z_i are components of u , y or x , then we get (a special case of) an ELCP.

3.3 The ELCP and hybrid systems

In Sections 3.1 and 3.2 we have shown that the ELCP arises in the analysis of certain classes of discrete event systems and continuous variable systems. Since hybrid systems arise from the interaction between discrete event systems and continuous variable systems, and since they exhibit characteristics of both discrete event systems and continuous variable systems, this leads to the question as to whether the ELCP can also play a role in the modeling and analysis of certain classes of hybrid systems. In the next section we shall show that this is indeed the case: we study a traffic-light-controlled intersection, which can be considered as a simple hybrid system. The evolution of the queue lengths in this hybrid system can be described by an ELCP.

Furthermore, in [19,21,22] Schumacher and van der Schaft consider another class of hybrid systems — the “complementary-slackness systems” — typical examples of which are electrical networks with diodes, or mechanical systems subject to geometric inequality constraints. They develop a method to determine the uniqueness of smooth continuations and to solve the associated mode selection problem for complementary-slackness systems. When the underlying system is a linear system, then this leads to a Linear Dynamic Complementarity Problem which can also be considered as a special case of the ELCP [11].

Hence, the ELCP can indeed be used in the analysis of certain classes of hybrid systems.

4 Traffic-light-controlled intersections

4.1 The set-up and the model of the system

Consider a single intersection of two two-way streets with controllable traffic lights on each corner (see Figure 2). For sake of brevity and simplicity we make the following assumptions:

- the traffic lights can either be red or green,
- the average arrival and departure rates of the cars are constant or slowly time-varying,
- the queue lengths are continuous variables.

These assumptions deserve a few remarks:

- Adding an all-red or amber phase leads to a similar, but more complex model (see [12]).
- If we keep in mind that one of the main purposes of the model that we shall derive, is the design of optimal traffic light switching schemes, then assuming that the average arrival and departure rates are constant is not a serious restriction, provided that we use a moving horizon strategy: we compute the optimal traffic light switching scheme for, say, the next 10 cycles, based on a prediction of the average arrival and departure rates (using data measured during the previous cycles) and we apply this scheme during the first of the 10 cycles, meanwhile we update our estimates of the arrival and departure rates and compute a new optimal scheme for the next 10 cycles, and so on.
- Designing optimal traffic light switching schemes is only useful if the arrival and departure rates of vehicles at the intersection are high. In that case, approximating the queue lengths by continuous variables only introduces small errors. Furthermore, in practice there is also some uncertainty and variation in time of the arrival and departure rates, which makes that in general computing the exact optimal traffic light switching scheme is utopian. Moreover, in practice we are more interested in quickly obtaining a good approximation of the optimal traffic light switching scheme than in spending a large amount of time to obtain the exact optimal switching scheme.

Let us now continue with the description of the set-up of the system. There are four lanes L_1, L_2, L_3 and L_4 , and on each corner of the intersection there are traffic lights (T_1, T_2, T_3 and T_4). The average arrival rate of cars in lane L_i is λ_i . When the traffic light is green, the average departure rate in lane L_i is μ_i . Let $t_0, t_1, t_2, t_3, \dots$ be the time instants at which the traffic lights switch from green to red or vice versa. The traffic light switching scheme is shown in Table 1. Define $\delta_k = t_{k+1} - t_k$. Let $l_i(t)$ be the queue length (i.e., the number of cars waiting) in lane L_i at time instant t .

Let us now write down the equations that describe the relation between the switching time instants and the queue lengths as continuous variables.

Consider lane L_1 . When the traffic light T_1 is red, there are arrivals at lane L_1 and no departures. As a consequence, we have

$$\frac{dl_1(t)}{dt} = \lambda_1 \quad (10)$$

Period	T_1	T_2	T_3	T_4
$t_0 - t_1$	red	green	red	green
$t_1 - t_2$	green	red	green	red
$t_2 - t_3$	red	green	red	green
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Table 1. The traffic light switching scheme.

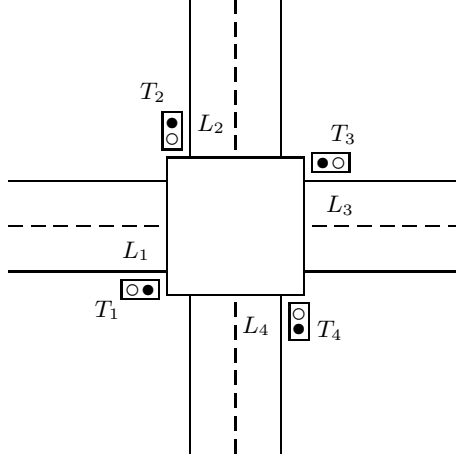


Fig. 2. A traffic-light-controlled intersection of two two-way streets.

for $t \in (t_{2k}, t_{2k+1})$ with $k \in \mathbb{N}$, and

$$l_1(t_{2k+1}) = l_1(t_{2k}) + \lambda_1 \delta_{2k}$$

for $k = 0, 1, 2, \dots$

When the traffic light T_1 is green, there are arrivals and departures at lane L_1 . Since then the net queue growth rate is $\lambda_1 - \mu_1$ and since the queue length $l_1(t)$ cannot be negative, we have

$$\frac{dl_1(t)}{dt} = \begin{cases} \lambda_1 - \mu_1 & \text{if } l_1(t) > 0 \\ 0 & \text{if } l_1(t) = 0 \end{cases} \quad (11)$$

for $t \in (t_{2k+1}, t_{2k+2})$ with $k \in \mathbb{N}$. So

$$l_1(t_{2k+2}) = \max(l_1(t_{2k+1}) + (\lambda_1 - \mu_1)\delta_{2k+1}, 0)$$

for $k = 0, 1, 2, \dots$

Note that we also have

$$l_1(t_{2k+1}) = \max(l_1(t_{2k}) + \lambda_1 \delta_{2k}, 0)$$

for $k = 0, 1, 2, \dots$ since $l_1(t) \geq 0$ for all t .

We can write down similar equations for $l_2(t_k)$, $l_3(t_k)$ and $l_4(t_k)$.

So if we define

$$x_k = \begin{bmatrix} l_1(t_k) \\ l_2(t_k) \\ l_3(t_k) \\ l_4(t_k) \end{bmatrix}, \quad b_1 = \begin{bmatrix} \lambda_1 \\ \lambda_2 - \mu_2 \\ \lambda_3 \\ \lambda_4 - \mu_4 \end{bmatrix}, \quad b_2 = \begin{bmatrix} \lambda_1 - \mu_1 \\ \lambda_2 \\ \lambda_3 - \mu_3 \\ \lambda_4 \end{bmatrix},$$

then we have

$$x_{2k+1} = \max(x_{2k} + b_1 \delta_{2k}, 0) \quad (12)$$

$$x_{2k+2} = \max(x_{2k+1} + b_2 \delta_{2k+1}, 0) \quad (13)$$

for $k = 0, 1, 2, \dots$

Remarks:

- The traffic-light-controlled intersection can be considered as a hybrid system that has the time and the queue lengths as state variables, and that can operate in two regimes characterized by differential equations of the form (10) or (11) depending on the value of a discrete control variable that can have the value “red” or “green”.
- The model we have derived is different from the models used by most other researchers due to the fact that we consider red-green cycle lengths that may vary from cycle to cycle. Furthermore, we consider non-saturated intersections, i.e., we allow queue lengths to become 0 during the green cycle. Some authors (see, e.g., [16,20]) only consider models for oversaturated intersections, i.e., they do not allow queue lengths to become equal to 0 during the green cycle. In that case the maximum operator that appears in (12) – (13) is not necessary any more, which leads to a simpler description of the behavior of the system. However, in [13] we have shown that, when we want to design optimal traffic light switching schemes, applying a model for oversaturated intersections to a non-saturated intersection in general does not lead to an optimal traffic light switching scheme.

4.2 Link with the ELCP

Let us now show that the system (12) – (13) can be reformulated as an ELCP. First consider (12) for an arbitrary index k . This equation can be rewritten as follows:

$$\begin{aligned} x_{2k+1} &\geq x_{2k} + b_1 \delta_{2k} \\ x_{2k+1} &\geq 0 \\ (x_{2k+1})_i &= (x_{2k} + b_1 \delta_{2k})_i \quad \text{or} \quad (x_{2k+1})_i = 0 \quad \text{for } i = 1, 2, 3, 4, \end{aligned}$$

or equivalently

$$\begin{aligned} x_{2k+1} - x_{2k} - b_1 \delta_{2k} &\geq 0 \\ x_{2k+1} &\geq 0 \\ (x_{2k+1} - x_{2k} - b_1 \delta_{2k})_i (x_{2k+1})_i &= 0 \quad \text{for all } i. \end{aligned}$$

Since a sum of nonnegative numbers is equal to 0 if and only if all the numbers are equal to 0, this system of equations is equivalent to:

$$x_{2k+1} - x_{2k} - b_1 \delta_{2k} \geq 0$$

$$x_{2k+1} \geq 0$$

$$\sum_{i=1}^4 (x_{2k+1} - x_{2k} - b_1 \delta_{2k})_i (x_{2k+1})_i = 0 \quad .$$

We can repeat this reasoning for (13) and for each index k .

If we consider N switching time instants and if we define

$$x^* = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} \quad \text{and} \quad \delta^* = \begin{bmatrix} \delta_0 \\ \delta_1 \\ \vdots \\ \delta_{N-1} \end{bmatrix} \quad ,$$

we finally get a description of the form

$$Ax^* + B\delta^* + c \geq 0 \tag{14}$$

$$x^* \geq 0 \tag{15}$$

$$(Ax^* + B\delta^* + c)^T x^* = 0 \quad . \tag{16}$$

It is easy to verify that the system (14)–(16) is a special case of an ELCP.

Now we can compute traffic light switching schemes that minimize objective functions such as

- (weighted) average queue length over all queues:

$$J_1 = \sum_{i=1}^4 w_i \frac{\int_{t_0}^{t_N} l_i(t) dt}{t_N - t_0} \quad ,$$

- (weighted) worst case queue length:

$$J_2 = \max_{i, t} (w_i l_i(t)) \quad ,$$

- (weighted) average waiting time over all queues:

$$J_3 = \sum_{i=1}^4 w_i \frac{\int_{t_0}^{t_N} l_i(t) dt}{\lambda_i(t_N - t_0)} \quad ,$$

and so on, where $w_i > 0$ for all i . Furthermore, we can impose extra conditions such as minimum and maximum durations for the green and the red time⁴, maximum queue lengths⁵, and so on. This leads to the following problem:

$$\text{minimize } J \tag{17}$$

⁴ A green time that is too short is wasteful. If the red time is too long, drivers tend to believe that the signals have broken down.

⁵ This could correspond to an upper bound on the available storage space due to the distance to the preceding junction or to the layout of the intersection.

subject to

$$\delta_{\min,r} \leq \delta_{2k} \leq \delta_{\max,r} \quad \text{for } k \in \alpha(N) \quad (18)$$

$$\delta_{\min,g} \leq \delta_{2k+1} \leq \delta_{\max,g} \quad \text{for } k \in \beta(N) \quad (19)$$

$$x_k \leq x_{\max} \quad \text{for } k = 1, 2, \dots, N \quad (20)$$

$$x_{2k+1} = \max(x_{2k} + b_1 \delta_{2k}, 0) \quad \text{for } k \in \alpha(N) \quad (21)$$

$$x_{2k+2} = \max(x_{2k+1} + b_2 \delta_{2k+1}, 0) \quad \text{for } k \in \beta(N) , \quad (22)$$

with

$$\alpha(N) = \left\{ 0, 1, \dots, \left\lfloor \frac{N-1}{2} \right\rfloor \right\} \quad \text{and} \quad \beta(N) = \left\{ 0, 1, \dots, \left\lfloor \frac{N}{2} \right\rfloor - 1 \right\} ,$$

where $\lfloor x \rfloor$ is the largest integer that is less than or equal to x .

It can be shown [13] that the objective function J_2 (i.e., the (weighted) worst case queue length) is convex as a function of the δ_k 's, which implies that problem (17)–(22) with $J = J_2$ can be solved efficiently (if there is no upper bound on the queue lengths, or if we deal with constraint (20) by introducing a convex penalty term if some components of $x_{\max}$ are finite). However, the objective functions J_1 and J_3 are neither convex nor concave. Our computational experiments have shown that in order to solve problem (17)–(22) with $J = J_1$ or $J = J_3$ using constrained optimization (with, e.g., sequential quadratic programming) several initial starting points are necessary to obtain the global minimum.

Using the procedure given above the system (18)–(22) can be rewritten as a system of the form

$$Ax^* + B\delta^* + c \geq 0 \quad (23)$$

$$x^* \geq 0 \quad (24)$$

$$Ex^* + D\delta^* + f \geq 0 \quad (25)$$

$$(Ax^* + B\delta^* + c)^T x^* = 0 , \quad (26)$$

which is again a special case of an ELCP. In order to determine the optimal traffic light switching scheme we could first determine the solution set of the ELCP and then minimize the objective function J over this solution set. Our computational experiments have shown that the determination of the minimum value of the objective functions J_1 and J_3 is a well-behaved problem in the sense that using a local minimization routine (that uses, e.g., sequential quadratic programming) starting from different initial points always yields the same numerical result (within a certain tolerance). Furthermore, it can be shown [13] that J_2 is a convex function of the parameters λ_k , κ_k and μ_k that characterize the solution set of the ELCP (cf. (2)).

The algorithm of [6,7] to compute the solution set of a general ELCP requires exponential execution times. This implies that the approach sketched above is not feasible if the number of switching cycles N is large. However, in [12] we have developed efficient methods to determine suboptimal traffic light switching schemes for the model (23)–(26): for the objective functions J_1 (i.e., (weighted)

average queue length) and J_3 (i.e., (weighted) average waiting time) we can make some approximations that transform the problem into an optimization problem over a convex feasible set, or even into a linear programming problem. This approach is computationally very efficient and yields suboptimal solutions that approximate the global optimal solution very well.

For more information on other models that describe the evolution of the queue lengths at a traffic-light-controlled intersection and on optimal traffic light control the interested reader is referred to [14,17,18,20] and the references given therein.

5 Conclusions and further research

We have introduced the Extended Linear Complementarity Problem (ELCP) and indicated how it can be used in the modeling and analysis of certain classes of discrete event systems, continuous variable systems and hybrid systems. More specifically, we have shown that for a traffic-light-controlled intersection the evolution of the queue lengths at the switching time instants can be described by an ELCP.

Topics for further research include: development of efficient algorithms for the special cases of the ELCP that appear in the analysis of specific classes of hybrid systems, investigation of the use of the ELCP to model and to analyze other classes of hybrid systems, and extension of our model for a traffic-light-controlled intersection to networks of intersections.

Acknowledgments

This research was sponsored by the Concerted Action Project of the Flemish Community, entitled “Model-based Information Processing Systems” (GOA-MIPS), by the Belgian program on interuniversity attraction poles (IUAP P4-02 and IUAP P4-24), by the Belgian sustainable mobility program “Traffic Congestion Problems in Belgium: Mathematical Models, Simulation, Control and Actions” (MD/01/24), and by the ALAPEDES project of the European Community Training and Mobility of Researchers Program.

Bart De Schutter is a senior research assistant of the F.W.O. (Fund for Scientific Research – Flanders) and Bart De Moor is a research associate with the F.W.O.

References

1. F. Baccelli, G. Cohen, G.J. Olsder, and J.P. Quadrat, *Synchronization and Linearity*. New York: John Wiley & Sons, 1992.
2. G. Cohen, D. Dubois, J.P. Quadrat, and M. Viot, “A linear-system-theoretic view of discrete-event processes and its use for performance evaluation in manufacturing,” *IEEE Transactions on Automatic Control*, vol. 30, no. 3, pp. 210–220, Mar. 1985.
3. R.W. Cottle, J.S. Pang, and R.E. Stone, *The Linear Complementarity Problem*. Boston: Academic Press, 1992.

4. R.A. Cuninghame-Green, *Minimax Algebra*, vol. 166 of *Lecture Notes in Economics and Mathematical Systems*. Berlin, Germany: Springer-Verlag, 1979.
5. B. De Moor, *Mathematical Concepts and Techniques for Modelling of Static and Dynamic Systems*. PhD thesis, Faculty of Applied Sciences, K.U.Leuven, Leuven, Belgium, June 1988.
6. B. De Schutter, *Max-Algebraic System Theory for Discrete Event Systems*. PhD thesis, Faculty of Applied Sciences, K.U.Leuven, Leuven, Belgium, Feb. 1996.
7. B. De Schutter and B. De Moor, "The extended linear complementarity problem," *Mathematical Programming*, vol. 71, no. 3, pp. 289–325, Dec. 1995.
8. B. De Schutter and B. De Moor, "Minimal realization in the max algebra is an extended linear complementarity problem," *Systems & Control Letters*, vol. 25, no. 2, pp. 103–111, May 1995.
9. B. De Schutter and B. De Moor, "Applications of the extended linear complementarity problem in the max-plus algebra," in *Proceedings of the International Workshop on Discrete Event Systems (WODES'96)*, Edinburgh, UK, pp. 69–74, Aug. 1996.
10. B. De Schutter and B. De Moor, "A method to find all solutions of a system of multivariate polynomial equalities and inequalities in the max algebra," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 6, no. 2, pp. 115–138, Mar. 1996.
11. B. De Schutter and B. De Moor, "The Linear Dynamic Complementarity Problem is a special case of the Extended Linear Complementarity Problem," Tech. rep. 97-21, ESAT-SISTA, K.U.Leuven, Leuven, Belgium, Mar. 1997. Accepted for publication in *Systems & Control Letters*, vol. 34, no. 1-2, pp. 63–75, June 1998.
12. B. De Schutter and B. De Moor, "Optimal traffic light control for a single intersection," Tech. rep. 96-90, ESAT-SISTA, K.U.Leuven, Leuven, Belgium, Jan. 1998. Revised version. Accepted for publication in *European Journal of Control*.
13. B. De Schutter and B. De Moor, "Optimal traffic light control for a single intersection: Addendum," Tech. rep. 96-90a, ESAT-SISTA, K.U.Leuven, Leuven, Belgium, Jan. 1998.
14. N.H. Gartner, J.D.C. Little, and H. Gabbay, "Simultaneous optimization of offsets, splits, and cycle time," *Transportation Research Record*, vol. 596, pp. 6–15, 1976.
15. S. Gaubert, *Théorie des Systèmes Linéaires dans les Dioides*. PhD thesis, Ecole Nationale Supérieure des Mines de Paris, France, July 1992.
16. J.H. Lim, S.H. Hwang, I.H. Suh, and Z. Bien, "Hierarchical optimal control of oversaturated urban traffic networks," *International Journal of Control*, vol. 33, no. 4, pp. 727–737, Apr. 1981.
17. D. Lin, F. Ulrich, L.L. Kinney, and K.S.P. Kumar, "Hierarchical techniques in traffic control," in *Proceedings of the IFAC/IFIP/IFORS 3rd International Symposium*, pp. 163–171, 1976.
18. E.S. Park, J.H. Lim, I.H. Suh, and Z. Bien, "Hierarchical optimal control of urban traffic networks," *International Journal of Control*, vol. 40, no. 4, pp. 813–829, Oct. 1984.
19. J.M. Schumacher, "Some modeling aspects of unilaterally constrained dynamics," in *Proceedings of the ESA International Workshop on Advanced Mathematical Methods in the Dynamics of Flexible Bodies*, ESA-ESTEC, Noordwijk, The Netherlands, June 1996.
20. M.G. Singh and H. Tamura, "Modelling and hierarchical optimization for oversaturated urban road traffic networks," *International Journal of Control*, vol. 20, no. 6, pp. 913–934, 1974.

21. A.J. van der Schaft and J.M. Schumacher, "Hybrid systems described by the complementarity formalism," in *Proceedings of the International Workshop on Hybrid and Real-Time Systems (HART'97)*, Grenoble, France, vol. 1201 of *Lecture Notes in Computer Science*, pp. 403–408, Springer-Verlag, 1997.
22. A.J. van der Schaft and J.M. Schumacher, "Complementarity modeling of hybrid systems," *IEEE Transactions on Automatic Control*, vol. 43, no. 4, pp. 483–490, Apr. 1998.
23. L. Vandenberghe, B. De Moor, and J. Vandewalle, "The generalized linear complementarity problem applied to the complete analysis of resistive piecewise-linear circuits," *IEEE Transactions on Circuits and Systems*, vol. 36, no. 11, pp. 1382–1391, Nov. 1989.

Controller Scheduling Using Neural Networks: Implementation and Experimental Results

Enrique D. Ferreira * and Bruce H. Krogh **

Department of Electrical and Computer Engineering
Carnegie Mellon University
5000 Forbes Av.
Pittsburgh, PA, 15213-3890 USA

edf/krogh@ece.cmu.edu

Abstract. This paper presents the results of simulation and control experiments using a recently proposed method for real-time switching among a pool of controllers. The switching strategy selects the current controller based on neural network estimates of the future system performance for each controller. This neural-network-based switching controller has been implemented for a simulated inverted pendulum and a level control system for an underwater vehicle in our laboratory. The objectives of the experiments presented here are to demonstrate the feasibility of this approach to switching control for real systems and to identify techniques to deal with practical issues that arise in the training of the neural networks and the real-time switching behavior of the system. This experimental work complements on-going theoretical investigations of the method which will be reported elsewhere.

1 Introduction

Recently there has been an interest in switching control schemes for adaptive control [1,2,8,9], and fault-tolerant control [10]. A switching control scheme based on neural network estimates of performance indices for the candidate controllers was proposed in [5]. The objectives of theoretical investigations of switching controllers are to establish sufficient conditions for closed-loop stability and bounds on closed-loop performance. This paper reports on empirical investigations of the switching control scheme proposed in [5] and considers some of the practical issues that arise in the implementation of switching control schemes. The basic rule of the switching strategy is to select the controller at each sampling time with the lowest performance index, as estimated by the ensemble of neural networks. This strategy is motivated by the so-called min-switch rule for multiple Lyapunov functions [4,7]. Each of the performance indices being estimated is a cost-to-go function which, under mild assumptions, would be a Lyapunov

* Research supported by a CONICYT-IBD grant from the government of Uruguay, and by the Organization of American States under grant F44395.

** Research supported by DARPA under contract F33615-97-C-1012.

function for the system under control of the corresponding state-feedback law. If these Lyapunov functions were known precisely, the min-switching strategy would lead to an asymptotically stable system. The strategy investigated in this paper addresses the practical issue of how to implement the min-switching strategy for real systems. For real systems, the true dynamics and Lyapunov functions are never known exactly. We demonstrate empirically that neural networks can be used to estimate cost-to-go functions effectively, and that acceptable closed-loop asymptotically stable behavior can be achieved using these estimates. This paper focuses on performance-based switching. The complete switching strategy includes an estimate of the region of stability for each controller; a controller is not available if the current state is outside the region of stability for the closed-loop system under that controller. Results related to the estimation of regions of stability using neural networks are presented in [6]. Typically this min-switching rule avoids selecting controllers for which the state of the system is outside the stability region, however. Since the performance index is very large (possibly infinite) for the controllers for which the state is outside the stability region, we found the stability region estimates to be unnecessary in the experiments reported in this paper. The paper is organized as follows. The next section presents a real-time infrastructure for fault-tolerant control as a primary motivation for the switching control problem considered in this paper. Two applications are also introduced for which multiple controllers have been designed and the problem is to establish switching rules to select the appropriate controllers in real time. Section 3 presents the neural-network-based switching strategy. Section 4 describes the method for training the neural networks to estimate closed-loop performance based on the concept of neural dynamic programming. Data is presented for training the neural networks for the two example applications. We then consider the effectiveness of the switching control strategy in section 5 where experimental results are presented for the two experimental examples. It is observed that chattering can occur due to the errors in the neural network estimates and methods for eliminating the chattering are discussed. The concluding section summarizes our experience with the neural-network-based switching control and identifies several directions for further investigations, both experimental and theoretical.

2 Applications of Switching Control Strategies

Switching control strategies are generally considered for situations where there are multiple operating regimes for a system and different controllers are designed for each operating regime. Switching control has also been proposed as an approach to adaptive control to deal with unknown plants. In this paper we consider a version of the former scenario where different controllers are available to stabilize a given system, but with different performance objectives. One controller might be designed to provide good transient performance for small deviations from the equilibrium, whereas another controller may be designed to drive the system quickly back to the equilibrium when large deviations are detected. The latter controller may perform poorly when the system is near the equilibrium,

so it is desirable to switch back to the first controller at some point. In this section we first describe a real-time infrastructure that relies on switching control strategies to provide fault tolerance against errors in untested control code. This architecture provides a practical motivation for the switching control strategy proposed in [5] and is the platform used for the control experiments presented in this paper. We then present two applications for which multiple controllers are available as test cases for the empirical investigations.

2.1 SIMPLEX Architecture

One of the main motivations for developing the controller switching strategy presented in this paper is to provide a method for implementing the switching rules in the SIMPLEX architecture, a technology developed at the Software Engineering Institute at Carnegie Mellon University to support safe, reliable on-line upgrades and modifications to real-time control systems [10]. Figure 1 shows a basic configuration for SIMPLEX. It consists of three controllers: a *safety* controller, a reliable *baseline* controller, and the *experimental* controller representing a new untested control module. The basic idea of the SIMPLEX system is to guarantee that the baseline controller performance is maintained if there are problems with the experimental controller. This is accomplished by monitoring the control outputs and system performance when the experimental controller is operating on the system, and switching control back to the baseline controller if problems are detected. The safety controller is invoked when it is necessary to take more extreme action to return the system to the operating region for the baseline controller. The controllers run concurrently as separate tasks in a real-time multitasking operating system, or even on different machines. SIMPLEX is in charge of all the communication between the parts of the system and monitors the performance of the controllers acting on the plant to determine which controller should actually be controlling the plant at each instant. The SIMPLEX architecture has been applied successfully to the control of sev-

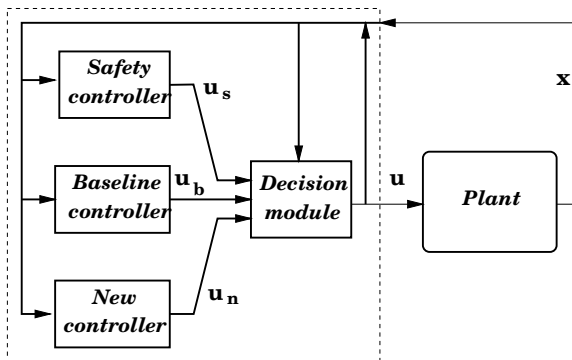


Fig. 1. SIMPLEX architecture.

eral physical systems including single and multiple inverted pendulum systems, a plasma deposition system, and a level-control system for a submerged vessel. We present two of these systems in the following subsections. From a control perspective, SIMPLEX corresponds precisely to the multi-controller scenario considered in this paper. The controllers are designed for a given plant with similar control objectives, in most cases to stabilize the system, but with different regions of stability and different performance characteristics. Clearly the ability for the SIMPLEX system to provide the desired protection against errors in the experimental controller depends entirely on the rules used to switch to the safety and baseline controllers. These rules are very difficult to determine analytically, even for low-dimensional systems. In the current applications of SIMPLEX, physical insight and extensive experimentation have been used to create ad hoc rules for switching to the safety and baseline controllers. The neural network approach proposed in this paper provides a means for determining automatically, and with some theoretical motivation, when to switch to the safety controller and the baseline controller. The on-line learning capabilities of the neural networks also suggest the possibility of acquiring knowledge of the performance of an experimental controller, leading to the eventual transition of the experimental controller to the new baseline controller once it has been completely tested and verified.

2.2 An Inverted Pendulum

As one example of an application of controller scheduling we use the well known inverted pendulum (IP) with a set of three controllers. One reason for using this application is that a physical IP has served as a standard laboratory example for the SIMPLEX architecture. The complete IP model and control design is described in [5]. We include here a brief summary. The IP model equations used are:

$$J_t \ddot{x} + \frac{1}{2} ml \cos \theta \ddot{\theta} + B_r \dot{x} - \frac{1}{2} ml \sin \theta \dot{\theta}^2 = F$$

$$\frac{1}{2} m \cos \theta \ddot{x} + \frac{1}{3} ml \ddot{\theta} - \frac{1}{2} mg \sin \theta = 0$$

with the following constraints:

$$|F| \leq F_{max}, \quad |x| \leq x_{max}, \quad |\dot{x}| \leq v_{max}$$

where θ is the angle of the pendulum, x is the position of the “cart” (the pendulum base), m is the effective mass at the end of the pendulum, l is the pendulum length, J_t is the inertia of the cart, B_r is the coefficient of friction for the cart, and g is gravitational acceleration. The parameters for the physical system in the laboratory are:

J_t	0.6650	Kg	g	9.8	m/s^2
m	0.21	Kg	F_{max}	2	N
l	0.61	m	x_{max}	2.0	m
B_r	0.1	Kg/s	v_{max}	3.0	m/s

The controllers designed for the system are: a standard linear quadratic regulator (LQR) controller for the linearized model of the IP around the origin in the full state space, a velocity feedback (VF) controller that takes the pendulum angle θ and the velocity of the cart to zero (but ignores the cart position), and a sliding mode (SM) controller designed to bring θ to zero as fast as possible using a nonlinear switching rule and saturating control (but without regard to the cart position or the cart speed). The need for a switching strategy is clear. There are different controllers with different characteristics available for a physical system. Given the designs of these three controllers, one would expect the performance of the SM controller to be best for larger pendulum angles, the VF controller to work well when the position of the cart is large, and the LQR controller to provide the best performance for small deviations from the origin of the four-dimensional state space. Therefore, there is a need for a supervisor to schedule those controllers in the best way possible, where "best" means asymptotic stability with good transient performance.

2.3 Submerged Vessel

This application is an experimental system in our laboratory consisting of a water tank in which a vessel ("diver") can move vertically by changing the size of the air bubble inside it. Air is moved in and out of the vessel through a flexible tube connected to a cylinder-piston mechanism. Figure 2 illustrates the system components and Figure 3 lists the physical parameters for the system. The position of the vessel and the size of the air bubble are measured directly using ultrasound sensors. A stepper motor controls the piston movement. The control objective is to stabilize the vessel at an arbitrary level without allowing the vessel to touch the bottom of the tank or emerge above the surface of the water. Any equilibrium position for the vessel is open-loop unstable since the air bubble compresses(expands) as the vessel goes down(up). There is also a significant time delay in the response of the vessel level to changes in the cylinder position due to the volume of the tube connecting the piston with the vessel. The following model has been developed for the system using first principles,

$$\begin{aligned}\ddot{y} &= -a|\dot{y}|\dot{y} - bh - f_r(\dot{y}) \\ \dot{h} &= -\alpha(y, h) - \beta(y, h)u(t - \tau)\end{aligned}$$

with

$$\begin{aligned}f_r(\dot{y}) &= f_1\dot{y} + f_2 \operatorname{sgn}(\dot{y}) e^{-f_3|\dot{y}|} \\ \alpha(y, h) &= \frac{\alpha_1}{\alpha_2 + (\alpha_3 + \alpha_y y + \alpha_h h)^2} \\ \beta(y, h) &= \frac{(\beta_1 + \beta_y y + \beta_h h)^2}{\alpha_2 + (\alpha_3 + \alpha_y y + \alpha_h h)^2}\end{aligned}$$

where y is the position of the vessel with respect to the water level and h is the height of the bubble. For the first equation governing y , the position of the

vessel, the first term of the right hand side is the water resistance, the second term reflects Archimedes' principle, and the last term is the Newtonian static and dynamic friction. The controlled input is the velocity of the piston that pushes the air between the vessel and the cylinder and τ is the time delay. State constraints are imposed by the bottom of the tank and the water level. The control input is limited by the maximum speed of the stepper motor. For

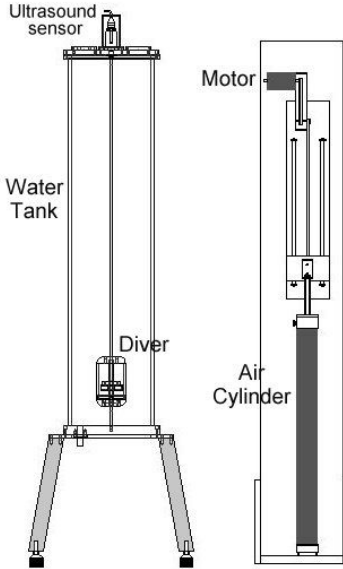


Fig. 2. Sketch of the components of the submerged vessel system.

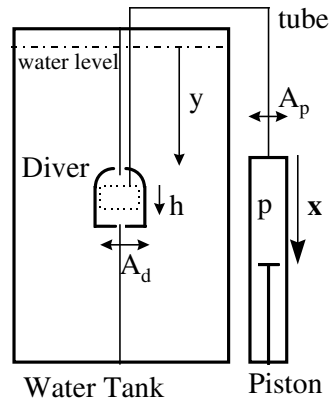


Fig. 3. Physical parameters for the submerged vessel system.

this example two linear state feedback controllers are available to regulate the position of the vessel. One of them is a bang-bang controller (BB) designed to operate when the vessel position is far from the desired position. The other controller is a linear state-feedback controller (LSF) that operates best when the vessel is close to the setpoint. The first controller drives the vessel to the setpoint much more quickly than the second controller, but the vessel oscillates around the setpoint in a limit cycle if the first controller is applied indefinitely. The scheduler must select which controller to apply at each instant to drive the vessel to the setpoint with the best performance, which includes some measure of the speed at which the vessel reaches the equilibrium and the quality of the regulation once the vessel is near the setpoint.

3 Neural Network Based Switching Strategy

As a general formulation of the problem being considered, let a state-constrained and control-constrained nonlinear system in its state space formulation be given by

$$\begin{aligned} \mathbf{x}_{k+1} &= f(\mathbf{x}_k, \mathbf{u}_k) \\ \mathbf{x}_k &\in S, \quad \mathbf{u}_k \in D \end{aligned} \quad (1)$$

where $\mathbf{x}_k \in R^n$ is the state vector and $\mathbf{u}_k \in R^m$ the control input vector. The nonlinear function f is assumed to be smooth. The discrete-time state equations reflect the sampled-data nature of the computer control system. We assume that the state $\mathbf{x}_k$ is observable. The control signal $\mathbf{u}_k$ to be applied on the system can be generated by M different state feedback controllers of the form:

$$\mathbf{u}_k^i = g_i(\mathbf{x}_k), \quad i = 1, 2, \dots, M. \quad (2)$$

The goal of the controllers is to take the system to the origin. We are interested in developing a strategy to select which controller should be applied to the system at each sampling instant in order to achieve stable response with good transient performance. In a similar setting there have been some results reported concerning the stability of a switching controller. Branicky [4] has established conditions for a switched autonomous system to be stable in the sense of Lyapunov using the concept of Lyapunov-like functions. Malmberg [7] examines asymptotic stability and explained chattering as a possible behavior generated intrinsically by the switching and not just as an implementation problem. To define our scheduling rule we need to define a performance index for each autonomous system generated by the application of a specific controller. Let us denote the trajectory of the system (1) with the initial state $\mathbf{x}_o$ at time step k by the application of the state feedback control $\mathbf{u}_k^i$ as $\mathbf{x}_k^i(\mathbf{x}_o)$. We consider performance indices for the controllers of the form

$$J_\delta^i(\mathbf{x}_o) = B^i + \sum_{k=0}^{\infty} \delta^k U^i(\mathbf{x}_k^i(\mathbf{x}_o)), \quad i = 1, \dots, M \quad (3)$$

where $0 < \delta < 1$ is a discount factor and $U(\cdot)$ represents the cost function of being in a particular state. The proposed approach for selecting the controller at each sampling instant is illustrated in Figure 4. Neural networks are used to estimate the performance indices J_δ^i at the current state, denoted by $\hat{J}_\delta^i$. The index of the control input to be applied for the next period, denoted i_k , is then selected as

$$i_k = \arg \min_{i \in I(\mathbf{x}_k, L_k)} \{ \hat{J}_\delta^i(\mathbf{x}_k) \} \quad (4)$$

where

$$I(\mathbf{x}_k, L_k) = \{ i | i = i_{k-1} \text{ or } i \neq i_{k-1}, \hat{J}_\delta^i(\mathbf{x}_k) < L_k^i \} \quad (5)$$

with $L_k = \{L_k^1, \dots, L_k^M\}$, and for $i = 1, \dots, M$

$$L_k^i = \begin{cases} L_{k-1}^i & \text{if } i \neq i_k \\ \min\{\hat{J}_\delta^i(x_k), L_{k-1}^i\} & \text{if } i = i_k \end{cases} \quad (6)$$

Expressed in words the switching rule will select the controller with the best performance but from a restricted set of controllers called $I(\mathbf{x}_k, L_k)$ consisting of the current controller and all the other controllers which have a better performance estimate than the last time they were used. This selection rule is related to the conditions stated by Branicky [4] to have stability in the sense of Lyapunov. Figure 4 illustrates how the scheduling rule works. When the estimate of the performance index of another controller becomes the lowest one and it is also lower than the last time that controller was applied – represented by the variable L_k^i – the switching rule selects that controller. An augmented version of this rule may include the stability region for each controller as an additional requirement for it to be in the set $I(\mathbf{x}_k, L_k)$. It has been pointed out above that this was not necessary for the applications in this paper.

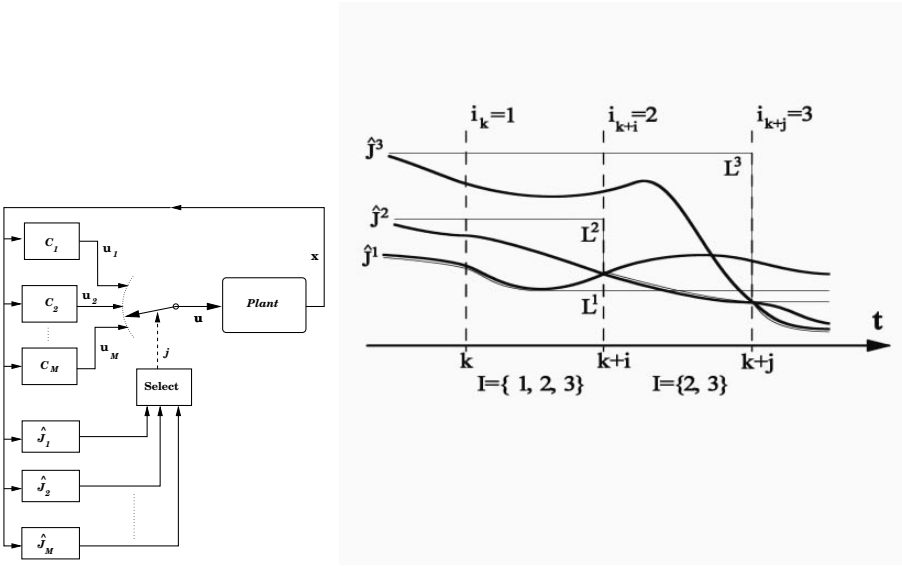


Fig. 4. Controller switching scheme used and example to show how it works.

4 Neural Network Training

This section describes a neural network architecture and training techniques to estimate the index of performance for a given controller. The proposed approach

is based on a modification of a dynamic programming procedure called neurodynamic programming (NDP) [3]. In NDP, a neural network is used to estimate the cost-to-go function. One motivation for using a neural network is for data compression: a neural network is much smaller than, say, a table representation of the cost-to-go function. The other motivation for using neural networks is the availability of techniques to train them to approximate the cost-to-go functions using data from the system. To estimate the performance indices $\hat{J}_\delta^i(\mathbf{x})$ defined in section 3 we chose neural networks of the form:

$$\hat{J}_\delta^i(\mathbf{x}) = \mathbf{b}_o + W_o \mathbf{x} + W_3 \phi(W_2 \phi(W_1 \mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2), \quad (7)$$

where $W_i, \mathbf{b}_i$ are the weight matrices and threshold vectors, respectively. The nonlinear function $\phi(\cdot)$ for the hidden units of the neural network is chosen to be the hyperbolic tangent function. The output units are selected as linear. Figure 5 shows a diagram of the network architecture.

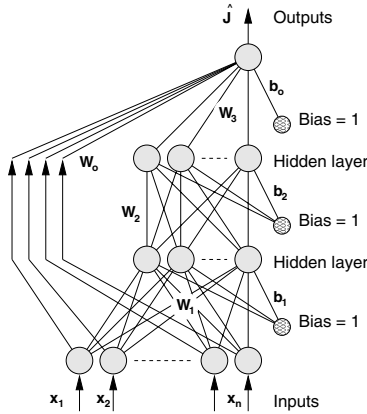


Fig. 5. Multilayer neural network.

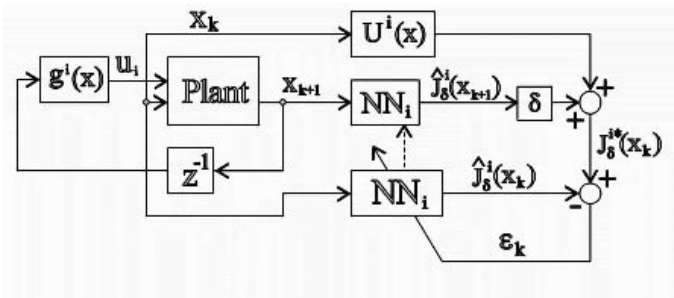


Fig. 6. HDP learning procedure diagram

Neuro-dynamic programming requires a type of learning procedure that is different from the well known supervised learning algorithms. We use Heuristic Dynamic Programming (HDP) algorithm [13]. If we denote by $\hat{J}_i(\mathbf{x})$ the network estimate of the future performance for the i th controller, the learning equation takes the form

$$J_\delta^{*i}(\mathbf{x}_k) = U(\mathbf{x}_k) + \delta \hat{J}_\delta^i(\mathbf{x}_{k+1}) \quad (8)$$

where $J_\delta^{*i}(\mathbf{x}_k)$ is the desired value for the network for state $\mathbf{x}_k$. The pointwise cost function selected is a standard quadratic form:

$$U(\mathbf{x}) = \mathbf{x}^T P \mathbf{x}, \quad P > 0$$

Using this cost function for each trajectory, the input-output patterns for the network can be computed. An error function for the network can be evaluated and a minimization procedure applied afterwards to adapt the parameters of the neural network. Figure 6 represents the learning procedure schematically. With this procedure we characterize each autonomous nonlinear system resulting from the application of a single controller. In this sense it is different from the Q-learning approach [12] that points to the design of a new controller by generating a performance index surface resulting from the collective action of all the controllers available at each state. In the case of the inverted pendulum example, three neural networks were trained to approximate the cost-to-go function of each closed-loop system generated by closing the loop with the three controllers. Using the simulation model, training was performed off-line in each case using 5000 random trajectories starting at states uniformly distributed over the allowable set of states. A set of 50 trajectories was used as a validation set. On-line learning is continued during the simulated control experiments using HDP. Equation (8) with $\delta = 0.5$, is used to generate the input-output patterns for the network. A square error function is computed and conjugate gradient optimization methods were used to minimize it. Off-line training was stopped when the error was 0.02 to prevent overtraining. Figures 7 and 8 show the estimation achieved for two controllers in a slice of the state space. The shapes are expected, close to a quadratic function near the origin and reaching large values outside its neighborhood. There is also some symmetry in the surface that can be explained from the model and cost functions symmetry. For the submerged vessel system two multilayer networks were set up to learn the performance index for the two controllers. In this case, 20 experiments were run on the physical system for each controller to get the data for training the networks. It should be pointed out that the experiments run on the vessel were selected with starting points along the region in which we were expecting the system to operate. Five additional experimental runs were saved for network validation purposes. After that, the procedure is similar to the previous example. Overtraining criteria stopped training when the overall error was about 0.035. Figures 9 and 10 show the estimation achieved for the two controllers designed for this example in a slice of the state space. Even though the error is bigger than in the previous example the shapes of the cost-to-go estimates have similar characteristics to the ones found for the pendulum example.

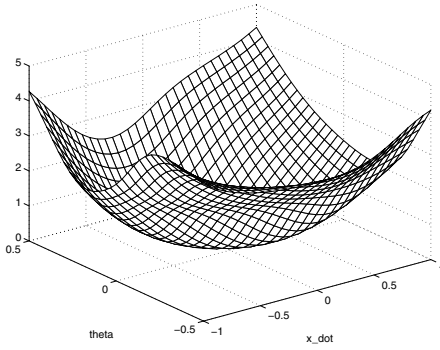


Fig. 7. Performance estimate for LQR controller. ($x = \dot{\theta} = 0$)

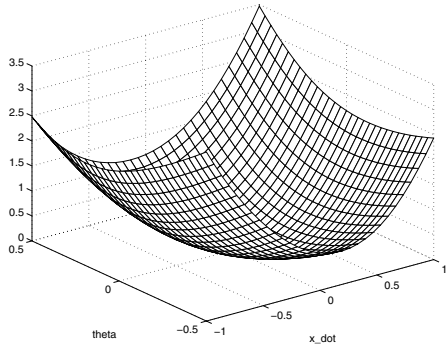


Fig. 8. Performance estimate for S-M controller. ($x = \dot{\theta} = 0$)

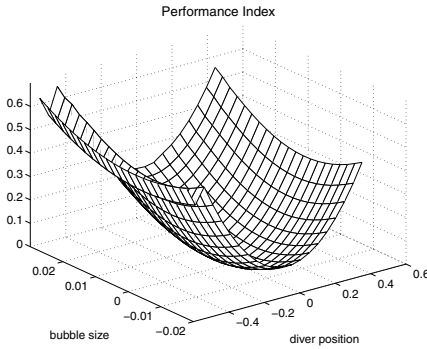


Fig. 9. Performance estimate for controller LSF. ($\dot{y} = 0$)

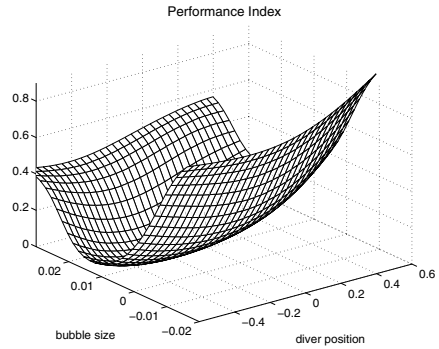


Fig. 10. Performance estimate for controller BB. ($\dot{y} = 0$)

5 Real-Time Control Results

Using the IP model described in section 2.2 we tested the scheduling policy defined by equations (4), (5) and (6) with simulation. Figures 11 and 12 show the position of the pendulum and the evolution of the performance index estimates during a typical run. We see in Figure 12 that the evolutions of the cost-to-go estimates are not monotonically decreasing. This is due to the fact that these are estimates of the true cost-to-go functions. Moreover, the discounted cost is not guaranteed to be a Lyapunov function for the system. In Figure 11, a slight chatter in the response of the switching mechanism can be observed. This is commonly observed in switching systems. It is caused by the discontinuity in the control signal when the system goes through the switching boundary and the new control value returns the system to the switching surface. For a strategy like the one we are using in this work, but in which the performance index is a Lyapunov function for each controller in closed loop with the system, it has been

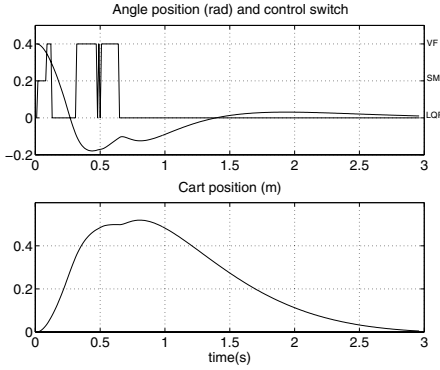


Fig. 11. Switching experiment

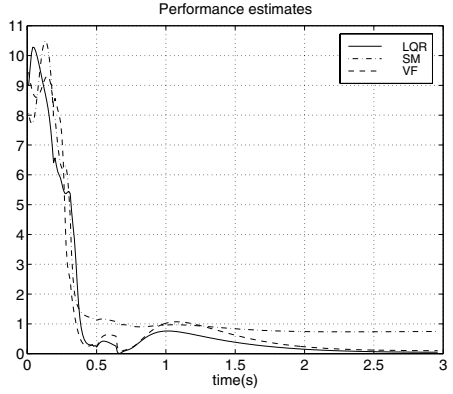


Fig. 12. Performance index for each controller during run.

shown [7] that in certain cases a chattering control drives the system along a surface in the state space where two or more Lyapunov functions are equal. This is commonly referred to as a sliding mode. When estimates are used, chattering can also be caused by the nonmonotonicity of the performance indices. There are several ways in which chattering has been eliminated or at least moderated before. The introduction of a boundary layer with many possible variations is a common approach [11]. Another way is the introduction of a non-zero minimum time between switches to reduce high frequency switching dynamics [2]. Another interesting idea is to compute the control that gives you the direction of the sliding surface as a way to eliminate the non-smooth behavior. The first two approaches can be applied in our method while the latter would not always be possible due to the usual lack of a system model. However, in this experiment the switching rule takes care of a big part of the possible chatter not allowing a controller to be used until its performance index has returned below its lowest previous value. For the underwater vessel control system, experimental runs were performed. Figure 13 shows a switching experiment for a step change in the setpoint value for y_{st} of 12 inches. Figure 5 shows the estimated performance indices during the run. Chatter is not observed in this example. From the Figure 5 we observe that the bang-bang controller is preferred for larger values of y . After approximately 5.5 seconds $\hat{J}_\delta^1$ becomes smaller than $\hat{J}_\delta^2$ and the scheduler switches to the linear state-feedback controller. In Figure 13 we see also an offset in the final position of the vessel due to the high static friction in the vertical bar along which the vessel slides. Other controllers are being designed to take care of that problem and can be introduced in the scheduling policy directly.

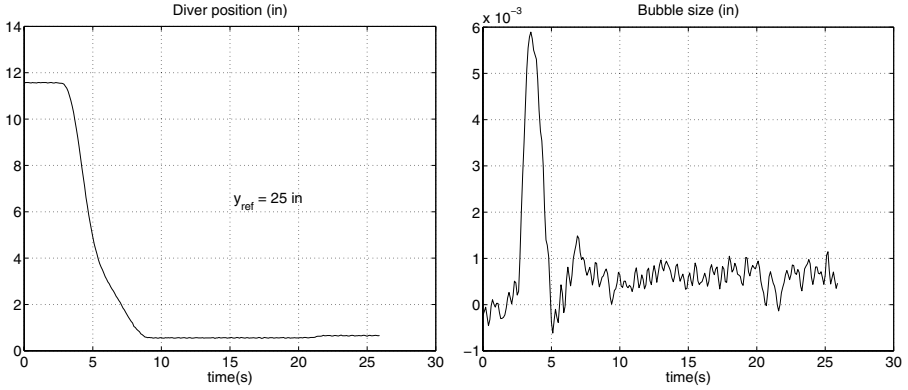


Fig. 13. Diver position and bubble size of the vessel during a switching experiment.

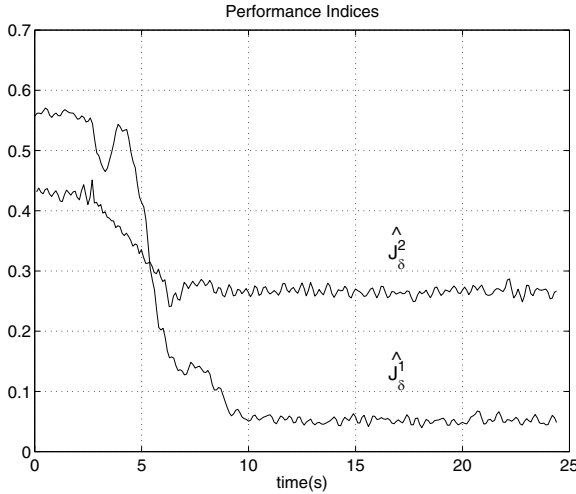


Fig. 14. Performance indices estimates for controllers for the submerged vessel during a switching experiment.

$\hat{J}_\delta^1$ - LSF controller. $\hat{J}_\delta^2$ - BB controller.

6 Summary and Discussions

A technique is presented to perform controller scheduling using a multilayer feedforward neural network and neuro-dynamic programming to estimate a performance index for each controller. Results from two applications are presented demonstrating the feasibility of the approach. Current research is focusing on learning algorithms to reduce the effect of the estimation error on the closed loop behavior. We are also investigating the use of better features as inputs to the

network and different network architectures to speed up learning. Furthermore, looking at the overall picture, comparisons are being made with other switching schemes like model-based Lyapunov function scheduling techniques in terms of stability, smoothness and robustness range. Finally, there are several theoretical issues currently under investigation. A recent result by Malmborg et al. [7] on the stability of the so-called *min-switching* rule based on Lyapunov stability theory suggests a way to analyze this architecture. Preliminary work in this area has been done but it remains under investigation to guarantee the convergence for the neuro-estimation learning algorithms and stability of the closed-loop system with adaptive controller scheduling.

References

1. J. Balakrishnan and K.S. Narendra. Adaptation and learning using multiple models, switching, and tuning. *IEEE Control Systems Magazine*, 15(3):37–51, Jun 1995.
2. J. Balakrishnan and K.S. Narendra. Adaptive control using multiple models. *IEEE Trans. on Automatic Control*, 42(2):171–187, Feb 1997.
3. Dimitri P. Bertsekas and John Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, 1996.
4. Michael S. Branicky. Stability of switched and hybrid systems. In *Proc. 33rd IEEE Conf. Decision Control*, volume 4, pages 3498–3503, Lake Buena Vista, FL, Dec 1994.
5. E.D. Ferreira and B.H. Krogh. Controller scheduling by neural networks. In *Proc. 36th Conf. Decision Control*, San Diego, CA, Dec 1997.
6. E.D. Ferreira and B.H. Krogh. Using neural networks to estimate regions of stability. In *Proc. of 1997 American Control Conference*, volume 3, pages 1989–93, Albuquerque, NM, Jun 1997.
7. J. Malmborg, B. Berhardsson, and K.J. Aström. A stabilizing switching scheme for multi-controller systems. In *Proc. of the IFAC World Congress*, volume F, pages 229–234, San Francisco, California, USA, 1996. IFAC'96, Elsevier Science.
8. A.S. Morse. Supervisory control of families of linear set-point controllers - Part II: Robustness. In *34th IEEE Conference on Decision and Control*, volume 2, pages 1750–5, New York, USA, Dec 1995.
9. A.S. Morse. Supervisory control of families of linear set-point controllers - Part I: Exact matching. *IEEE Trans. on Automatic Control*, 41(10):1413–31, Oct 1996.
10. L. Sha. A software architecture for dependable and evolvable industrial computing systems. In *Proc. IPC'95*, pages 145–156, Detroit, MI, May 1995.
11. Jean-Jacques. E. Slotine and Weiping Li. *Applied Nonlinear Control*. Prentice Hall Inc., 1991.
12. C.J.C.H. Watkins. *Learning from Delayed Rewards*. PhD thesis, Cambridge University, Cambridge, England, 1989.
13. P. Werbos. A menu of designs in reinforcement learning over time. In W.T. Miller III, R.S. Sutton, and P. Werbos, editors, *Neural Networks for control*, chapter 3. MIT Press, 2nd edition, 1991.

Rigorous Design of a Fault Diagnosis and Isolation Algorithm

Jianping Gao^{*} and Qiwen Xu

International Institute for Software Technology
The United Nations University
P.O.Box 3058, Macau
Email: qxu@iist.unu.edu

Abstract. In this paper, we present a rigorous design of a Fault Diagnosis and Isolation algorithm. The system is modelled as a hybrid system with a network of parallel components. The requirement is specified in Duration Calculus, a dense time temporal logic. We use traditional program logic, suitably extended, to verify the discrete component and subsequently derive a number of properties of the system. Finally, the requirement is shown to be satisfied by proving that it can be deduced from the system properties.

1 Introduction

In virtually all control systems, some physical variables must be measured. In the real world, the sensors which perform this job may be faulty, either temporarily or permanently. Therefore, to achieve a high degree reliability, several copies of sensors are used to sample the same variable. By comparing the sampled values from the sensors, and possibly some analytic reasoning based on the dynamical behaviours of the plant, a monitor tries to identify a sensor which is not faulty and uses its output as the final sampled value.

Usually, a Fault Diagnosis and Isolation (FDI) system contains an odd number of sensors, and the output is chosen to be one agreed upon by the majority. However, there are cases that an even number sensors (which cannot produce a majority when exactly half of the sensors are faulty) have to be used. For example, it could be that the sensor is very costly, or there is not enough space to install yet another sensor. In this paper, we study an FDI system with four sensors. As illustrated by the following diagram, the variable to be sampled is y , and output y_v is decided by the monitor based on four inputs y_1 to y_4 from the sensors.

^{*} On leave from Department of Automatic Control, Beijing University of Aeronautics and Astronautics, Beijing, 100083, P.R. China. Email: gjp@ns.dept3.buaa.edu.cn

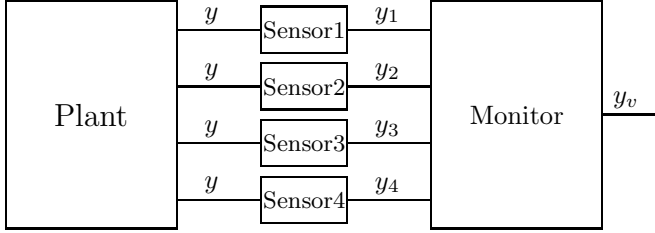


Figure 1: An FDI system.

Typically, an FDI system should satisfy the following two properties: 1, the accumulated time that the final output is not correct should be less than a certain fraction of the total elapsed time; 2, permanent faults should be identified with related sensors isolated. To offer a high level assurance, we need to reason rigorously that the design indeed satisfies the requirements besides the traditional validation methods of simulation and testing.

An FDI can be considered as a hybrid system with interacting continuous and discrete components. The output of a sensor is the summation of the nominal output, the noise function and the fault function. The nominal output is continuous obeying certain dynamical laws, the noise function is relatively small and the fault function is discrete reflecting instantaneous occurrence of faults.

In the literature, hybrid systems are often represented as Automata or as Phase Transition Systems [8,7]. In our approach, we work with a notation for hybrid systems which is closer to how hybrid systems are programmed. In particular, complex digital control algorithms can be directly incorporated in our framework as subprograms. Therefore, we call the notation Hybrid Programs. As an effective way to verify a hybrid program, it is suggested in [13] to use a program logic to verify the discrete component, and incorporate the results into a real-time temporal logic which is subsequently used to deduce properties of the whole system. This approach has been further studied in [12,11].

This paper is organised as follows. In section 2, we briefly review the Duration Calculus (DC) [5,6], which is a dense time temporal logic based on the Interval Temporal Logic (ITL) [10]. We next introduce the notation of Hybrid Program, and the verification method for it. The requirement of the FDI system is analysed in section 4. The system is modelled as a hybrid program in section 5 and verified in section 6.

2 Duration Calculus

In this paper, the calculus we use is based on several variants of DC [5,6,4], but we shall still call it DC for short.

DC was developed to reason about piece-wise continuous functions of time called states, which model the status of the system. In DC, time is represented

by non-negative reals. Intervals and interpretations of state variables are defined as follows

$$\text{Intv} \stackrel{\text{def}}{=} \{[c, d] \in \text{Time} \times \text{Time} \mid c \leq d\}$$

$$\mathcal{I} \in SVar \rightarrow \text{Time} \rightarrow \text{Values}.$$

Roughly speaking, a model is a pair $(\mathcal{I}, [c, d])$ of interpretation and interval. A state expression is constructed from state variables with appropriate operators. A Boolean state expression, such as $x < 1$, with values represented by 0 and 1, denotes a property of the system. The duration that a Boolean state expression B holds in a model $(\mathcal{I}, [c, d])$ is defined as

$$\llbracket \int B \rrbracket(\mathcal{I}, [c, d]) \stackrel{\text{def}}{=} \int_c^d \llbracket B \rrbracket(\mathcal{I}, t) dt$$

where $\llbracket B \rrbracket(\mathcal{I}, t)$ denotes the value of B at time t under state interpretation $\mathcal{I}$. The length l of an interval is defined as

$$l \stackrel{\text{def}}{=} \int 1$$

and it is easy to prove

$$\llbracket l \rrbracket(\mathcal{I}, [c, d]) = d - c.$$

For any state expression S , we define its limit values at the beginning and the end of a non-empty interval as

$$(\mathbf{b}.S)(\mathcal{I}, [c, d]) \stackrel{\text{def}}{=} \lim_{t \rightarrow c^+} \llbracket S \rrbracket(\mathcal{I}, t),$$

$$(\mathbf{e}.S)(\mathcal{I}, [c, d]) \stackrel{\text{def}}{=} \lim_{t \rightarrow d^-} \llbracket S \rrbracket(\mathcal{I}, t).$$

A point interval is characterised by $l = 0$, shortened as $\lceil \cdot \rceil$. A Boolean state expression B holds everywhere inside an interval and at a point are denoted respectively by $\lceil B \rceil$ and $\lceil B \rceil^0$, and are formally defined as

$$(\mathcal{I}, [c, d]) \models \lceil B \rceil \stackrel{\text{def}}{=} c < d \text{ and for any } c < t < d, \llbracket B \rrbracket(\mathcal{I}, t) = 1$$

$$(\mathcal{I}, [c, d]) \models \lceil B \rceil^0 \stackrel{\text{def}}{=} c = d \text{ and } \llbracket B \rrbracket(\mathcal{I}, c) = 1.$$

Let $\lceil B \rceil^* \stackrel{\text{def}}{=} \lceil B \rceil \vee \lceil B \rceil^0$. The modality ‘chop’ of ITL is defined as follows: for any formulae A and B

$$(\mathcal{I}, [c, d]) \models A; B$$

iff there exists m such that $c \leq m \leq d$ and

$$(\mathcal{I}, [c, m]) \models A \text{ and } (\mathcal{I}, [m, d]) \models B.$$

Two modalities derived from chop are

$$\Diamond A \stackrel{\text{def}}{=} \text{true}; A; \text{true}$$

$$\Box A \stackrel{\text{def}}{=} \neg \Diamond \neg A.$$

A model satisfies $\Diamond A$ and $\Box A$ if respectively a sub-interval and all sub-intervals satisfy A . Both $\Diamond$ and $\Box$ are contracting modalities in the sense that they only refer to sub-intervals. The following modality allows us to access intervals that are left neighbourhoods of the original interval

$$(\mathcal{I}, [c, d]) \models \Diamond_l A \quad \text{iff} \quad \text{there exists } \delta \geq 0 \text{ such that } (\mathcal{I}, [c - \delta, d]) \models A$$

The dual notion of right neighbourhood also exists, but we do not need it for our case study.

The axiomatic system of DC includes that of ITL and the following axioms and rules. The axioms about durations are

$$\begin{aligned} \int 0 &= 0 \\ \int 1 &= l \\ \int B &\geq 0 \\ \int B_1 + \int B_2 &= \int (B_1 \vee B_2) + \int (B_1 \wedge B_2) \\ ((\int B = x); (\int B = y)) &\Rightarrow (\int B = x + y) \\ \int B_1 = \int B_2, &\text{ provided } B_1 \Leftrightarrow B_2 \text{ holds in propositional logic.} \end{aligned}$$

Mathematical theories can be imported into the calculus by the following MT rule

Let $R([H(S)], \mathbf{b}.S, \mathbf{e}.S, l)$ be a formula without chop and neighbourhood modalities. If in mathematics $(\forall c \leq d) R(\forall c \leq t \leq d. H(S(t)), S(c^+), S(d^-), d - c)$ holds, then $\Box \vee R([H(S)], \mathbf{b}.S, \mathbf{e}.S, l)$ holds in DC,

where $S(c^+)$ and $S(d^-)$ denote limits of S from the right of c and left of d respectively.

The following are some examples of DC formulae

$(\int x > 1) < 0.25l$	duration that $x > 1$ holds is less than a quarter of the length of the interval
$[x < 1] \wedge l > 3$	$x < 1$ holds everywhere inside the interval and the length is greater than 3
$([x < 1] \wedge l > 3); ([x > 5] \wedge l = 1)$	the interval can be divided in two parts satisfying respectively $[x < 1] \wedge l > 3$ and $[x > 5] \wedge l = 1$

and theorems

$$\begin{aligned} ([x < 1] \wedge l > 3); [x = 0]^0; ([x > 5] \wedge l = 1) &\Rightarrow [x < 1 \vee x > 5] \wedge l > 4 \\ ([x < 1] \wedge l > 3); l = 1 &\Rightarrow (\int x > 1) < 0.25l \\ \mathbf{b}.y < 5 \wedge [x < 1 \wedge \dot{y} = x] \wedge l < 3 &\Rightarrow [y < 8]. \end{aligned}$$

The proof system also contains two induction rules which are somewhat complex. In the last step of our case study, we need the following one

Backward Induction Rule. Let $R(X)$ be a formula without the neighbourhood operator. If $R(\top)$ and $R(X) \Rightarrow R(X \vee (\top B]; X) \vee (\top \neg B]; X))$ hold, then $R(\text{true})$ holds.

3 Hybrid Programs

3.1 The Language

In [12], a language for describing sequential hybrid systems was proposed. The language supports constructive design by allowing specification statements to be mixed with program constructs, in the tradition of the well established theories of program development such as refinement calculus [2,9]. In addition to usual program constructs, the language includes a specification statement for describing (instantaneous) state transitions and a specification statement using a DC formula for describing a possibly hybrid component with time dependent dynamics. To describe concurrency, we extend the language with a parallel composition.

The abstract syntax and informal interpretation of the basic commands are as follows.

$$\begin{aligned} P &::= \langle B(\bar{x}) \rightarrow R(\bar{x}, \bar{x}') \rangle \mid \ll C \gg \mid P_1; P_2 \mid P_1 \sqcap P_2 \mid \text{while } b \text{ do } P \text{ od} \\ S &::= P_1 \parallel P_2 \parallel \cdots \parallel P_n. \end{aligned}$$

The first statement is the guarded nondeterministic assignment: the process waits until the Boolean condition B holds, and then the statement is successfully executed and effect is an instantaneous transition satisfying the binary predicate R . Sequential composition is represented by $P_1; P_2$ and its meaning is that if P_1 terminates then P_2 is executed immediately afterwards. Nondeterministic choice is represented by $\sqcap$. Statement $\ll C \gg$ describes a component by a DC formula C . Continuous evolution can be specified by an invariant expressing the dynamical laws. Iteration statement is executed repeatedly until the boolean guard becomes false. Parallel composition is defined by the interleaving of component transitions.

Other commands can be constructed from the above primitive ones, e.g.,

$$\begin{aligned} \bar{x} &:= \text{exp}(\bar{x}) \stackrel{\text{def}}{=} \langle \text{true} \rightarrow \bar{x}' = \text{exp}(\bar{x}) \rangle \\ \text{skip} &\stackrel{\text{def}}{=} x := x \\ \langle B(\bar{x}) \rangle &\stackrel{\text{def}}{=} \langle \text{true} \rightarrow B(\bar{x}) \rangle \\ \text{if } B_1 \rightarrow P_1 \sqcap \cdots \sqcap B_n \rightarrow P_n \text{ fi} &\stackrel{\text{def}}{=} \langle B_1 \rangle; P_1 \sqcap \cdots \sqcap \langle B_n \rangle; P_n \end{aligned}$$

3.2 Verification of Hybrid Programs

Verification methods for discrete programs, such as Hoare Logic, have been well established and can be found in text books (e.g., [1]). As an effective way to verify a hybrid program, it is suggested in [13] to use a program logic to verify

the discrete component, and incorporate the results into a real-time temporal logic which is subsequently used to deduce properties of the whole system. For our case study, we extend the method to handle a limited form of interference that exists among processes.

Sequential Discrete Programs In Hoare logic, a program S is correct with respect to a pair of pre- and post-conditions is denoted by

$$\{p\} S \{q\}.$$

The meaning is that if the program is started in a state satisfying p , it will terminate in a state satisfying q . For example, the following is a valid correctness formula

$$\{x = 0\} x := x + 1 \{x = 1\}.$$

The following are some rules in Hoare Logic

$$\frac{\{p\} S_1 \{r\} \quad \{r\} S_2 \{q\}}{\{p\} S_1; S_2 \{q\}}$$

$$\frac{\{p\} S_1 \{q\} \quad \{p\} S_2 \{q\}}{\{p\} S_1 \parallel S_2 \{q\}}$$

$$\frac{\{p \wedge B_1\} S_1 \{q\} \quad \cdots \quad \{p \wedge B_n\} S_n \{q\}}{\{p\} \text{ if } B_1 \rightarrow P_1 \parallel \cdots \parallel B_n \rightarrow P_n \text{ fi } \{q\}}$$

$$\frac{\begin{array}{l} r(i) \wedge i > 0 \Rightarrow B \quad r(0) \Rightarrow \neg B \\ \{r(i) \wedge i > 0\} S \{(\exists j < i) r(j)\} \end{array}}{\{\exists i. r(i)\} \text{ while } B \text{ do } S \text{ od } \{r(0)\}}$$

$$\frac{p \Rightarrow p_1 \quad \{p_1\} S \{q_1\} \quad q_1 \Rightarrow q}{\{p\} S \{q\}}$$

Proofs of programs are usually presented by annotating the program with assertions which serve as the pre-condition for the statement following it and the post-condition for the statement proceeding it.

Linking Program Logic with DC We propose two rules linking program logic with DC

$$\frac{\{\text{true}\} S \{q(\bar{x})\}}{S; \ll C \gg \underline{\text{sat}} \quad q(\mathbf{b}.\bar{x}) \wedge C}$$

This allows the post-condition of a discrete program to be used as ‘pre-condition’ of the next phase.

$$\frac{C_1 \Rightarrow p(\mathbf{e}.\bar{x}) \quad \{p(\bar{x})\} S \{q(\bar{x})\}}{\ll C_1 \gg; S; \ll C_2 \gg \underline{\text{sat}} \quad C_1; (q(\mathbf{b}.\bar{x}) \wedge C_2)}$$

This rule says that the ‘post-condition’ of the first phase can be used as the pre-condition of the discrete program, and as in the rule before, the post-condition of the discrete program can be added as ‘pre-condition’ of the next phase. These rules allow us to reason the mixture of discrete and continuous components based on properties of the components.

Soundness of the rules follows from techniques developed in [12,11], where the semantics of sequential programs is defined using a more involved extension of DC. The calculus is based on the Super Dense Duration Calculus first suggested by Zhou and Hansen [3], and it can describe discrete transitions and their compositions. However, for practical applications, one can use ordinary DC and high-level rules like the ones presented above. For example, we have the following proof for the sequential program

$$\begin{array}{l} \{\text{true}\} \\ x := 0; \\ \{x = 0\} \\ x := x + 1; \\ \{x = 1\} \\ y := 0 \\ \{x = 1 \wedge y = 0\} \end{array}$$

A phase which lasts for one time unit and obeys dynamic laws $\dot{x} = 0$ and $\dot{y} \leq x$ is represented by $\ll [\dot{x} = 0 \wedge \dot{y} \leq x] \wedge l = 1 \gg$. It follows that the hybrid program

$$x := 0; x := x + 1; y := 0; \ll [\dot{x} = 0 \wedge \dot{y} \leq x] \wedge l = 1 \gg$$

satisfies

$$\mathbf{b}.x = 1 \wedge \mathbf{b}.y = 0 \wedge [\dot{x} = 0 \wedge \dot{y} \leq x] \wedge l = 1$$

and we can further deduce for example that it satisfies

$$[x = 1 \wedge y \leq 1]$$

Concurrency and Interference In general, it is difficult to verify concurrent programs which may interfere with each other. But for particular applications, such as our case study, the degree of interference is usually limited, and consequently it is possible to use special and hence effective methods to verify the system.

In our case study, the only shared variables are the sampled variables. They are read by the monitor and changed by the plant. The sampling action can be modelled by an assignment $x := \exp(y_1, \dots, y_n)$, where $\exp(y_1, \dots, y_n)$ is an expression involving variables $y_1, \dots, y_n$. If $y_1, \dots, y_n$ are continuous, then the Hoare triple

$$\{\text{true}\} \ x := \exp(y_1, \dots, y_n) \ \{x = \exp(y_1, \dots, y_n)\}$$

is a valid axiom. However, if some of the variables may be changed discretely, as the fault functions in our case study, the triple is not a valid axiom, because the variables can change at exactly the sampling points. In most cases, it is reasonable to assume that the sampled variables can change at most once at one time point, and under this assumption, we can conclude that the sampled values are either the values before or after the transition if it indeed occurs. Formally, we have the following axiom

$$\{\text{true}\} \ x := \exp(y_1, \dots, y_n) \ \{q\}$$

where q is the assertion $\{\exists z_1, \dots, z_n. x = e(y_1, \dots, y_n) \wedge (z_1 = y_1^- \vee z_1 = y_1) \wedge \dots \wedge (z_n = y_n^- \vee z_n = y_n)\}$, in which z_i is a fresh variable and y_i^- denotes the left limit of y_i .

The link rule can be generalised

$$\frac{C_1 \Rightarrow p(\mathbf{e}.\bar{x}) \quad \{p(\bar{x})\} \ S \ \{q(x_1, \dots, x_m, y_1^-, \dots, y_n^-)\}}{\ll C_1 \gg; S; \ll C_2 \gg \quad \underline{\text{sat}} \quad (\exists a_1, \dots, a_n. (C_1 \wedge \mathbf{e}.y_1 = a_1 \wedge \dots \wedge \mathbf{e}.y_n = a_n) ; (q(\mathbf{b}.x_1, \dots, \mathbf{b}.x_m, a_1, \dots, a_n) \wedge C_2))}$$

Consider the simple hybrid program

$$\ll C_1 \gg; x := y; u := x > 0; \ll C_2 \gg .$$

The system has two phases, and in between the value of y is sampled and afterwards u is set to true or false depending on whether the sample value is greater than 0. Variable y can be changed at most once by the environment at any point, whereas u is not changed by the environment. We have

$$\begin{aligned} &\{\text{true}\} \\ &x := y; \\ &\{\exists z. x = z \wedge (z = y^- \vee z = y)\} \\ &u := x > 0 \\ &\{(u \Leftrightarrow x > 0) \wedge (x = y^- \vee x = y)\} \end{aligned}$$

and it follows that

$$\ll C_1 \gg; x := y; u := x > 0; \ll C_2 \gg$$

satisfies

$$\exists a. (C_1 \wedge \mathbf{e}.y = a); ((\mathbf{b}.u \Leftrightarrow \mathbf{b}.x > 0) \wedge (\mathbf{b}.x = a \vee \mathbf{b}.x = \mathbf{b}.y) \wedge C_2).$$

Suppose $C_1 \Rightarrow l = T$ and $C_2 \Rightarrow l = T$, that is, each phase takes T time units. Furthermore, assume $C_2 \Rightarrow \text{stable}(u)$, indicating that in the second phase u is not changed. We can prove by using DC that the program satisfies

$$\Box((l > T \wedge \lceil u \rceil) \Rightarrow \Diamond(y > 0))$$

and

$$\Box((l > T \wedge \lceil y > 0 \rceil) \Rightarrow \text{true}; \lceil u \rceil).$$

4 System Analysis and Requirement Specification

4.1 Sensors and fault models

Under our model, the output of sensor i ($i = 1, 2, 3, 4$) at time t is

$$y_i(t) = y(t) + n_i(t) + f_i(t), \quad i = 1, 2, 3, 4,$$

where $y(t)$ is the nominal output, $n_i(t)$ is the noise function and $f_i(t)$ is the fault function. We have the following assumptions about these:

- The nominal output is continuous with $|\dot{y}(t)| < L$.
- The noise is smaller than a positive constant value N .
- Fault functions are discontinuous: $f_i(t) = 0$ when the i th sensor is not faulty, $|f_i(t)| \geq F$, where F is a positive constant when the i th sensor is faulty; faults occur instantaneously, that is, the values of fault functions can jump between the two kinds of values, but will not lie in anywhere between 0 and F .
- Fault values are substantially larger than noise values, and in particular, we assume $F > 4 * N$.

Usually, sensors will stay in one status, either faulty or non-faulty, for a while before changing into a different one. We assume that there exist at least T_1 time units between the changes of the status of any two sensors. Also, a sensor will not be faulty for a long time unless it is severely damaged. We assume that if a sensor has been faulty for t_p time units, it must be caused by serious damage and the sensor becomes permanently faulty.

We model sensor i as a hybrid program

```

 $S_i \hat{=}$ 
 $c_i := 0; c_d := 0; f_i := 0;$ 
while true do
  if  $c_i < t_p \wedge d > T_1 \wedge |f_i| \geq F \rightarrow d := 0; f_i := 0; c_i := 0; c_d := 0$ 
  []  $c_i < t_p \wedge d > T_1 \wedge f_i = 0 \rightarrow d := 0; \langle \text{true} \rightarrow |f'_i| \geq F \rangle; c_i := 0; c_d := 1$ 
  []  $\text{true} \rightarrow \ll [(y_i = y + n_i + f_i) \wedge \dot{f}_i = 0 \wedge n_i < N \wedge \dot{d} = 1 \wedge \dot{c}_i = c_d] \gg$ 
  fi
od

```

in which c_i and d (initially $d = 0$) count respectively the time that sensor i has been faulty and the time since the last change of any sensor status. The change of the status of sensor i from faulty to non-faulty and from non-faulty to faulty is represented respectively by assignment $f_i := 0$ and nondeterministic assignment $\langle \text{true} \rightarrow |f'_i| \geq F \rangle$. Continuous evolution of various variables related to the sensor is governed by laws in the phase statement.

It is easy to prove that the following invariants are satisfied

Property 1 $\square[|n_i| \leq N]^*, \square[|f_i| \geq F \vee f_i = 0]^*, i = 1, 2, 3, 4$

Let

$$F_i(t) = \begin{cases} 1 & |f_i(t)| \geq F \\ 0 & \text{otherwise} \end{cases}$$

Boolean function F_i holds if and only sensor i is faulty. Since change of sensor status can only happen if the sensor has not been faulty for more than t_p time units, ensured by the conjunct $c_i < t_p$ in the guard, we have the following property

Property 2 $\square((\lceil F_i \rceil \wedge l \geq t_p); \text{true} \Rightarrow \lceil F_i \rceil)$

Let

$$\begin{aligned} P_i &\triangleq (f_i^+ = 0 \wedge f_i^- = 0) \vee (|f_i^+| \geq F \wedge |f_i^-| \geq F) \\ P &\triangleq P_1 \wedge P_2 \wedge P_3 \wedge P_4 \end{aligned}$$

Boolean function P_i holds if and only if sensor i is not changing its status, and P holds if and only if none of the sensors are changing the status. The following formula expresses the property that between two changes, there is at least T_1 units time

Property 3 $\square(\lceil \neg P \rceil^0; \lceil P \rceil; \lceil \neg P \rceil^0 \Rightarrow l \geq T_1)$

This implies the following lemma.

Lemma 1

$$\left(\begin{array}{l} \square(\lceil F_i \rceil; (l \leq T); \lceil F_i \rceil \Rightarrow \lceil F_i \rceil) \\ \wedge \\ \square(\lceil \neg F_i \rceil; (l \leq T); \lceil \neg F_i \rceil \Rightarrow \lceil \neg F_i \rceil) \end{array} \quad i = 1, 2, 3, 4 \right)$$

Faults are typically independent, and in our model, we assume that if two sensors are faulty, the difference between the two fault functions are greater than a certain value G and $G > 4 * N$.

Assumption 1

$$(f_i \neq 0 \wedge f_j \neq 0) \Rightarrow |f_i - f_j| \geq G \text{ for any } i, j \in \{1, 2, 3, 4\}, i \neq j.$$

Under the normal condition, it is unlikely that three sensors are faulty, except the whole sensor area is damaged. We therefore assume that if there are three more faulty sensors, the fourth one is also faulty.

Assumption 2

$$\begin{aligned} (f_i \neq 0 \wedge f_j \neq 0 \wedge f_k \neq 0) &\Rightarrow f_l \neq 0 \\ \text{for any } i, j, k, l &\in \{1, 2, 3, 4\}, i \neq j \neq k \neq l. \end{aligned}$$

4.2 Requirement

Let

$$FFF(t) \stackrel{\text{def}}{=} \bigvee_{i \neq j \neq k} F_i(t) \wedge F_j(t) \wedge F_k(t).$$

Boolean function FFF holds if and only there are three (actually four by Assumption 2) faulty sensors. When this happens, it is impossible to use the sampled values in any way, and the final output will be assigned a default value FSV . Therefore, the required output is defined as follows

$$y_r(t) = \begin{cases} FSV & FFF(t) = 1 \\ y(t) & \text{otherwise.} \end{cases}$$

Due to delay in detecting fault occurrence, sometimes the actual output is different from the required one. Let y_v denote the actual output and define

$$D(t) = \begin{cases} 1 & |y_v(t) - y_r(t)| \leq \epsilon \\ 0 & |y_v(t) - y_r(t)| > \epsilon. \end{cases}$$

Boolean function D holds if and only if the actual output deviates from the required one by not more than a fixed amount ϵ . When D holds, we say that the system is under proper working condition.

The first requirement is that the system should stay in proper working condition sufficiently often. More precisely, we require that if P holds initially for more than T_1 time units, then the accumulated proper working time is not less than $100(1 - \delta)\%$ ($0 < \delta < 1$) of elapsed time. This is expressed in DC as

$$REQ_1 \stackrel{\text{def}}{=} \Box((\lceil P \rceil \wedge l > T_1); \text{true} \Rightarrow \int D \geq (1 - \delta) * l).$$

For this to be possible, the noise values cannot be too large. In particular, we assume

Assumption 3 $N \leq \epsilon/2$.

Permanent faults must be identified within a specified amount of time Δ . Let PF_i be the boolean function which holds if and only if sensor i is detected to be permanently faulty. The second requirement is

$$REQ_2 \stackrel{\text{def}}{=} \Box(\lceil F_i \rceil \wedge (l \geq t_p + \Delta) \Rightarrow \text{true}; \lceil PF_i \rceil).$$

The third requirement is that if a sensor is considered to be permanently faulty, it must have been faulty for at least t_p time units

$$REQ_3 \stackrel{\text{def}}{=} \Box(l \geq t_p; \lceil PF_i \rceil \Rightarrow \text{true}; (\lceil F_i \rceil \wedge l \geq t_p)).$$

5 Design of the Monitor

The system is modelled as the parallel composition of the sensors, a monitor and the plant.

$$SYS \triangleq S_1 \parallel S_2 \parallel S_3 \parallel S_4 \parallel M/P$$

Since the plant is very simple, we combine it with the monitor and denote them together by M/P .

Our main job is to design the monitor so that the requirements are satisfied. As an initial attempt, we have the following framework of the system

```

M/P  $\triangleq$ 
count( $i$ ) := 0;  $PF_i$  := false ( $i = 1, 2, 3, 4$ );
while true do
   $x_i$  :=  $y_i$  ( $i = 1, 2, 3, 4$ );
  if  $\neg PF_1 \wedge \neg PF_2 \wedge \neg PF_3 \wedge \neg PF_4 \rightarrow P_0$ ;
     $\square_{i,j,k,l} PF_i \wedge \neg PF_j \wedge \neg PF_k \wedge \neg PF_l \rightarrow P_1^i$ ;
     $\square_{i,j,k,l} PF_i \wedge PF_j \wedge \neg PF_k \wedge \neg PF_l \rightarrow P_2^{ij}$ ;
     $\square_{i,j,k} PF_i \wedge PF_j \wedge PF_k \rightarrow y_v := \text{FSV}$ ;
  fi
   $\ll [(\neg FFF \rightarrow y_r = y) \wedge (FFF \rightarrow y_r = \text{FSV}) \wedge |\dot{y}| < L] \wedge l = T \gg$ ;
   $PF_i := PF_i \vee \text{count}(i) \geq T_p$  ( $i = 1, 2, 3, 4$ );
od

```

Variable $\text{count}(i)$ records the time that sensor i has been detected faulty. Boolean variable PF_i , as introduced earlier, indicates whether sensor i is considered permanently faulty. It has the initial value false, and is set to true when the value of $\text{count}(i)$ exceeds a thrash-hold value T_p , which is a control parameter yet to be chosen.

Roughly, the system works as follows. At each point of sampling which has a cycle of T , the value of sensor i is sampled and stored in x_i . According to whether there are permanently faulty sensors, the following procedures P_0 , P_1^i and P_2^{ij} decide if a sensor is faulty by comparing the sampled values. If a sensor is detected to be faulty, then the counter associated with it is increased at the end of the phase. During the phase, faults may occur, and consequently the value of requested output y_r changes between the actual value y and FSV, with the latter being a default value.

Procedure P_0 covers the case that there are no permanent faults.

```

 $P_0 \hat{=}$ 
 $FC_i := (|x_i - x_j| \geq T_h) \wedge (|x_i - x_k| \geq T_h) \wedge (|x_i - x_l| \geq T_h) \ (i = 1, 2, 3, 4);$ 
if
 $\neg FC_1 \wedge \neg FC_2 \wedge \neg FC_3 \wedge \neg FC_4 \rightarrow$ 
 $count(m) := 0 \ (m = 1, 2, 3, 4); \ y_v := x_1;$ 
 $\square_{i,j,k,l} FC_i \wedge \neg FC_j \wedge \neg FC_k \wedge \neg FC_l \rightarrow$ 
 $count(i) := count(i) + T; count(j) := 0;$ 
 $count(k) := 0; count(l) := 0; y_v := x_k;$ 
 $\square_{i,j,k,l} FC_i \wedge FC_j \wedge \neg FC_k \wedge \neg FC_l \rightarrow$ 
 $count(i) := count(i) + T; count(j) := count(j) + T;$ 
 $count(k) := 0; count(l) := 0; y_v := x_k;$ 
 $\square_{i,j,k} FC_i \wedge FC_j \wedge FC_k \rightarrow$ 
 $count(m) := count(m) + T \ (m = 1, 2, 3, 4); \ y_v := FSV;$ 
fi

```

By comparing the sampled value of sensor i with the values of the rest three sensors, it is decided whether it is faulty and variable FC_i is assigned to true or false accordingly. This step is crucial and an appropriate thrash-hold value T_h must be chosen for the outcome of that decision to be meaningful. Based on the value of FC_i , variable $count(i)$ is updated, and output y_v is either assigned the value of a sensor which is considered non-faulty or the default value FSV if three sensors are regarded faulty. Procedures P_1^i and P_1^{ij} are similar, and they are given in the appendix.

As the design decision, the control parameters in the algorithm are chosen to satisfy the following

$$\begin{pmatrix} T_h = \min(F, G) - 2 * N \\ T_p = \lceil \frac{t_p}{T} \rceil * T + 2 * T \\ T < \min(\frac{\Delta}{3}, \frac{\epsilon}{2 * L}, \frac{T_1 * \delta}{2}). \end{pmatrix} \quad \text{DES}$$

In the next section, we argue that the requirements are indeed satisfied.

6 Verification

We divide verification into several steps. First, we verify the control algorithm using program logic. Afterwards, we extract a number of properties and express them in DC. Finally, we prove that the requirements are implied by the established properties. For brevity, we will usually not write out all the conditions explicitly, but refer to them in the proof when they are used. We shall write P1 for Property 1, A2 for Assumption 2 etc. in the proof when there is not enough space,

6.1 Verification of Control Algorithm

By the axiom in 3.2, we know that $x_i = y_i^- \vee x_i = y_i$ ($i = 1, 2, 3, 4$) is a valid postcondition of the assignment $x_i := y_i$. It can then be used as the precondition of P_0 . The following lemma shows that under this condition, if $|x_i - x_j| \geq T_h \wedge |x_i - x_k| \geq T_h \wedge |x_i - x_l| \geq T_h$ holds, then either the left or the right limit of $|f_i|$ is greater than or equal to F , that is, sensor i has been faulty before or after the sampling point.

Lemma 2

$$(x_m = y + n_m + f_m) \vee (x_m = y + n_m + f_m^-) \quad m = 1, 2, 3, 4 \\ \Rightarrow \left((|x_i - x_j| \geq T_h \wedge |x_i - x_k| \geq T_h \wedge |x_i - x_l| \geq T_h \Rightarrow f_i^- \neq 0 \vee f_i \neq 0) \right) \\ \Rightarrow (f_i = 0 \vee f_i^- = 0)$$

Proof:

$$\left(\begin{array}{l} ((x_m = y + n_m + f_m) \vee (x_m = y + n_m + f_m^-)) \quad m = 1, 2, 3, 4 \\ \wedge (|x_i - x_j| \geq T_h) \quad \text{for } j \neq i \\ \wedge \neg(f_i^- \neq 0 \vee f_i \neq 0) \end{array} \right) \{\text{DES}\} \\ \Rightarrow \left(\begin{array}{l} (|n_i - n_j + f_i - f_j| \geq \min(F, G) - 2 * N \\ \vee |n_i - n_j + f_i - f_j^-| \geq \min(F, G) - 2 * N \\ \vee |n_i - n_j + f_i^- - f_j| \geq \min(F, G) - 2 * N \\ \vee |n_i - n_j + f_i^- - f_j^-| \geq \min(F, G) - 2 * N) \quad \text{for } j \neq i \\ \wedge f_i^- = 0 \wedge f_i = 0 \end{array} \right) \{\text{P1}\} \\ \Rightarrow \left(\begin{array}{l} f_i^- = 0 \wedge f_i = 0 \\ \wedge |f_j^-| \geq \min(F, G) - 4 * N \\ \wedge |f_j| \geq \min(F, G) - 4 * N \quad \text{for } j \neq i \end{array} \right) \{F > 4 * N \text{ and } G > 4 * N\} \\ \Rightarrow \left(\begin{array}{l} f_i^- = 0 \wedge f_i = 0 \\ \wedge f_j^- \neq 0 \wedge f_j \neq 0 \quad \text{for } j \neq i \end{array} \right) \{\text{A2}\} \\ \Rightarrow \text{false.}$$

And

$$\left(\begin{array}{l} ((x_m = y + n_m + f_m) \vee (x_m = y + n_m + f_m^-)) \quad m = 1, 2, 3, 4 \\ \wedge \neg(|x_i - x_j| \geq T_h \wedge |x_i - x_k| \geq T_h \wedge |x_i - x_l| \geq T_h) \\ \wedge \neg(f_i = 0 \vee f_i^- = 0) \end{array} \right) \{\text{P1}\} \\ \Rightarrow \left(\begin{array}{l} ((x_m = y + n_m + f_m) \vee (x_m = y + n_m + f_m^-)) \quad m = 1, 2, 3, 4 \\ \wedge \neg(|x_i - x_j| \geq T_h \wedge |x_i - x_k| \geq T_h \wedge |x_i - x_l| \geq T_h) \\ \wedge f_i \geq F \wedge f_i^- \geq F \\ \wedge (|f_j^-| \geq F \vee |f_j| = 0) \quad \text{for } j \neq i \end{array} \right) \{\text{A1}\} \\ \Rightarrow \left(\begin{array}{l} ((x_m = y + n_m + f_m) \vee (x_m = y + n_m + f_m^-)) \quad m = 1, 2, 3, 4 \\ \wedge \neg(|x_i - x_j| \geq T_h \wedge |x_i - x_k| \geq T_h \wedge |x_i - x_l| \geq T_h) \\ \wedge |f_i - f_j| \geq \min(F, G) \wedge |f_i - f_j^-| \geq \min(F, G) \\ \wedge |f_i^- - f_j| \geq \min(F, G) \wedge |f_i^- - f_j^-| \geq \min(F, G) \quad \text{for } j \neq i \end{array} \right) \{\text{P1}\}$$

$$\begin{aligned}
&\Rightarrow \left(\neg(|x_i - x_j| \geq T_h \wedge |x_i - x_k| \geq T_h \wedge |x_i - x_l| \geq T_h) \right) \quad \{\text{DES}\} \\
&\Rightarrow \left(\begin{aligned} &\wedge |x_i - x_j| \geq \min(F, G) - 2 * N \quad \text{for } j \neq i \\ &\neg(|x_i - x_j| \geq T_h \wedge |x_i - x_k| \geq T_h \wedge |x_i - x_l| \geq T_h) \end{aligned} \right) \\
&\Rightarrow \left(\wedge |x_i - x_j| \geq T_h \quad \text{for } j \neq i \right) \\
&\Rightarrow \text{false.}
\end{aligned}$$

□

Let

$$Q_i \stackrel{\text{def}}{=} (FC_i \wedge |f_i| \geq F \wedge |f_i^-| \geq F) \vee (\neg FC_i \wedge f_i = 0 \wedge f_i^- = 0)$$

The following diagram illustrates the relationships among F_i , FC_i , Q_i and P_i .

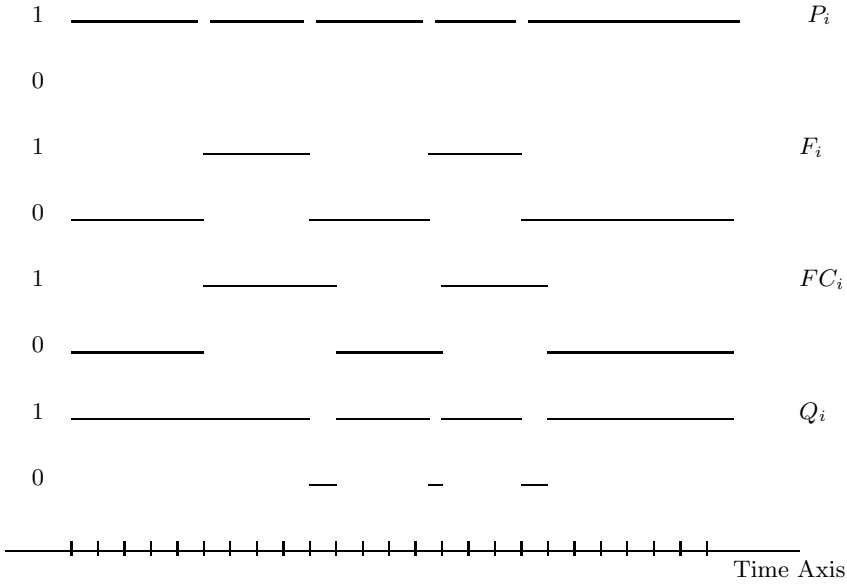


Figure 2: The relationships among F_i , FC_i , Q_i , P_i .

As shorthands, let

$$\begin{aligned}
Q &\stackrel{\text{def}}{=} Q_1 \wedge Q_2 \wedge Q_3 \wedge Q_4 \\
FFFC &\stackrel{\text{def}}{=} \bigvee_{i,j,k} FC_i \wedge FC_j \wedge FC_k, i, j, k \in \{1, 2, 3, 4\} \\
I(FC) &\stackrel{\text{def}}{=} \bigwedge_i ((\neg FC_i \wedge (f_i = 0 \vee f_i^- = 0)) \vee (FC_i \wedge (|f_i| \geq F \vee |f_i^-| \geq F))) \\
I(count) &\stackrel{\text{def}}{=} \bigwedge_i ((\neg FC_i \wedge count(i) = 0) \vee (FC_i \wedge count(i) = count(i)^- + T)) \\
I(FC, count) &\stackrel{\text{def}}{=} I(FC) \wedge I(count)
\end{aligned}$$

We have the following proof outline for procedure P_0 :

```

{ $x_m = y_m^- \vee x_m = y_m$ }
 $FC_i := (|x_i - x_j| \geq T_h) \wedge (|x_i - x_k| \geq T_h) \wedge (|x_i - x_l| \geq T_h) \ (i = 1, 2, 3, 4);$ 
 $\left\{ \begin{array}{l} (x_m = y_m^- \vee x_m = y_m) \wedge \\ (FC_i \Leftrightarrow (|x_i - x_j| \geq T_h) \wedge (|x_i - x_k| \geq T_h) \wedge (|x_i - x_l| \geq T_h)) \end{array} \right\}$ 
 $\{x_m = y_m^- \vee x_m = y_m\} \wedge I(FC) \}$  {Lemma 2}
if
 $\neg FC_1 \wedge \neg FC_2 \wedge \neg FC_3 \wedge \neg FC_4 \rightarrow$ 
 $\{ (x_m = y_m^- \vee x_m = y_m) \wedge I(FC) \wedge \neg FC_1 \wedge \neg FC_2 \wedge \neg FC_3 \wedge \neg FC_4 \}$ 
 $count(m) := 0 \ (m = 1, 2, 3, 4); y_v := x_1;$ 
 $\left\{ \begin{array}{l} (x_m = y_m^- \vee x_m = y_m) \wedge \\ I(FC, count) \wedge \neg FC_1 \wedge \neg FC_2 \wedge \neg FC_3 \wedge \neg FC_4 \wedge y_v = x_1 \end{array} \right\}$ 
 $\square_{i,j,k,l} FC_i \wedge \neg FC_j \wedge \neg FC_k \wedge \neg FC_l \rightarrow$ 
 $\{ (x_m = y_m^- \vee x_m = y_m) \wedge I(FC) \wedge FC_i \wedge \neg FC_j \wedge \neg FC_k \wedge \neg FC_l \}$ 
 $count(i) := count(i) + T; count(j) := 0;$ 
 $count(k) := 0; count(l) := 0; y_v := x_k;$ 
 $\left\{ \begin{array}{l} (x_m = y_m^- \vee x_m = y_m) \wedge \\ I(FC, count) \wedge FC_i \wedge \neg FC_j \wedge \neg FC_k \wedge \neg FC_l \wedge y_v = x_k \end{array} \right\}$ 
 $\square_{i,j,k,l} FC_i \wedge FC_j \wedge \neg FC_k \wedge \neg FC_l \rightarrow$ 
 $\{ (x_m = y_m^- \vee x_m = y_m) \wedge I(FC) \wedge FC_i \wedge FC_j \wedge \neg FC_k \wedge \neg FC_l \}$ 
 $count(i) := count(i) + T; count(j) := count(j) + T;$ 
 $count(k) := 0; count(l) := 0; y_v := x_k;$ 
 $\left\{ \begin{array}{l} (x_m = y_m^- \vee x_m = y_m) \wedge \\ I(FC, count) \wedge FC_i \wedge FC_j \wedge \neg FC_k \wedge \neg FC_l \wedge y_v = x_k \end{array} \right\}$ 
 $\square_{i,j,k} FC_i \wedge FC_j \wedge FC_k \rightarrow$ 
 $\{ (x_m = y_m^- \vee x_m = y_m) \wedge I(FC) \wedge FC_i \wedge FC_j \wedge FC_k \}$ 
 $count(m) := count(m) + T \ (m = 1, 2, 3, 4); y_v := FSV;$ 
 $\{ (x_m = y_m^- \vee x_i = y_i) \wedge I(FC, count) \wedge FC_i \wedge FC_j \wedge FC_k \wedge y_v = FSV \}$ 
fi
 $\{ I(FC, count) \wedge (Q \wedge \neg FFF_c \Rightarrow |y_v - y| \leq N) \wedge (FFF_c \Rightarrow (y_v = FSV)) \}$ 

```

The fact $Q \wedge \neg FFF_c \Rightarrow |y_v - y| \leq N$ is a valid post-condition relies on the following lemma

Lemma 3 $Q \wedge (x_k = y_k^- \vee x_k = y_k) \wedge \neg FC_k \Rightarrow |x_k - y| \leq N$

Proof:

$Q \wedge (x_k = y_k^- \vee x_k = y_k) \wedge \neg FC_k$	{Definition of Q }
$\Rightarrow (x_k = y_k^- \vee x_k = y_k) \wedge f_k^- = 0 \wedge f_k = 0$	{Definition of y_k }
$\Rightarrow x_k = y + n_k$	{Property 1}
$\Rightarrow x_k - y \leq N$	

□

We can similarly prove the same postcondition for procedures P_1^i, P_2^{ij} . It is easy to see that we can associate the following proof outline for M/P

```

M/P
count(i) := 0; PFi := false (i = 1, 2, 3, 4);
while true do
  xi := yi (i = 1, 2, 3, 4);
  {(xi = yi- ∨ xi = yi) (i = 1, 2, 3, 4)}
  if ¬PF1 ∧ ¬PF2 ∧ ¬PF3 ∧ ¬PF4 → P0;
    [ i,j,k,l PFi ∧ ¬PFj ∧ ¬PFk ∧ ¬PFl → P1i;
      [ i,j,k,l PFi ∧ PFj ∧ ¬PFk ∧ ¬PFl → P2ij;
        [ i,j,k PFi ∧ PFj ∧ PFk → yv := FSV;
          fi
        {I(FC, count) ∧ (Q ∧ ¬FFFc ⇒ |yv - y| ≤ N) ∧ (FFFc ⇒ (yv = FSV))}
        ≪ [ (¬FFF → yr = y) ∧ (FFF → yr = FSV) ∧ |ȳ| < L ] ∧ l = T ≫;
        PFi := PFi ∨ count(i) ≥ Tp (i = 1, 2, 3, 4);
      od

```

We next incorporate the results into DC, based partially on the link rules and partially on operational reasoning. The remarks following the stated property give some explanation why it holds.

Property 4 $\square([F_i] \wedge l > T \Rightarrow l \leq T; [FC_i])$

Remark: If F_i holds (i.e., sensor is faulty) for a period of time greater than T , then at the next sampling time, this will be detected with FC_i set to true.

Property 5 $\square(l \geq m * T \wedge [FC_i] \Rightarrow \diamond([F_i] \wedge l \geq (m - 1) * T))$

Remark: If FC_i holds for greater than or equal to $m * T$ time units, then F_i holds either in the left or the right neighbourhood intervals of the first $m * T$ sampling points (including the beginning point of the interval if it starts with a sampling point). Since the distance between two neighbourhood intervals is less or equal to T and $T < T_1$, it follows from Lemma 1 that F_i holds everywhere between the m sampling points.

Property 6 $\square(l > T \wedge [P] \Rightarrow l \leq T; [Q])$

Remark: If the status of a sensor is not changed, then after a sampling point, the detected status is the actual one and this is maintained in the rest of the interval.

Lemma 4

$$\square \left(([|y_v - y| \leq N]^0; \text{true}) \wedge [y_r = y \wedge |\dot{y}| < L] \wedge \text{stable}(y_v) \wedge (l \leq T) \right) \\ \Rightarrow [|y_r - y_v| < \epsilon]$$

where $\text{stable}(y_v)$ denote variable y_v does not change in the interval.

Proof:

$$\begin{aligned}
& (\lceil |y_v - y| \leq N \rceil^0; \text{true}) \wedge \lceil y_r = y \wedge |\dot{y}| < L \rceil \wedge \text{stable}(y_v) \wedge (l \leq T) \quad \{\text{A3}\} \\
\Rightarrow & (\lceil |y_v - y| \leq \frac{\epsilon}{2} \rceil^0; \text{true}) \wedge \lceil y_r = y \wedge |\dot{y}| < L \rceil \wedge \text{stable}(y_v) \wedge (l \leq T) \quad \{\text{MT}\} \\
\Rightarrow & \lceil |y_v - y| < \frac{\epsilon}{2} + L * T \rceil \wedge \lceil y_r = y \rceil \quad \{T < \frac{\epsilon}{2 * L}\} \\
\Rightarrow & \lceil |y_v - y_r| < \epsilon \rceil
\end{aligned}$$

□

Property 7
$$\begin{pmatrix}
\Box(\lceil Q \wedge \neg FFF_c \rceil \Rightarrow \lceil |y_v - y_r| \leq \epsilon \rceil) \\
\wedge \\
\Box(\lceil Q \wedge FFF_c \rceil \Rightarrow \lceil y_v = y_r = \text{FSV} \rceil)
\end{pmatrix}$$

Remark: It follows from Lemma 1 and the Lemma 4 above.

Property 8
$$\Box(\lceil FC_i \rceil \wedge (l > T_p) \Rightarrow (\text{true}; \lceil PF_i \rceil))$$

Property 9
$$\Box(\lceil PF_i \rceil \Rightarrow \Diamond_l(\lceil FC_i \rceil \wedge l \geq T_p))$$

Remark: Since $I(\text{count})$ holds before the phase is entered, it follows that if FC_i holds at the sampling point, then the associated counter will be increased. Therefore, if FC_i holds longer than T_p , then the counter will reach the value T_p causing PF_i to be true. On the other hand, if PF_i is set to true, FC_i must hold longer than T_p .

6.2 Verification in DC

We now deduce requirements REQ_1 , REQ_2 and REQ_3 in DC.

Lemma 5
$$\Box(\lceil FC_i \rceil \wedge l \geq T_p \Rightarrow (\text{true}; (\lceil F_i \rceil \wedge l \geq t_p)))$$

Proof:

$$\begin{aligned}
& \lceil FC_i \rceil \wedge l \geq T_p \quad \{T_p = \lceil \frac{t_p}{T} \rceil * T + 2 * T, \text{Property 5}\} \\
\Rightarrow & \Diamond(\lceil F_i \rceil \wedge (l \geq \lceil \frac{t_p}{T} \rceil * T + T)) \quad \{\text{Arithmetics}\} \\
\Rightarrow & \text{true}; (\lceil F_i \rceil \wedge l \geq t_p); \text{true} \quad \{\text{Assumption 2}\} \\
\Rightarrow & \text{true}; (\lceil F_i \rceil \wedge l \geq t_p)
\end{aligned}$$

□

Theorem 1
$$\begin{pmatrix}
\Box(\lceil F_i \rceil \wedge (l \geq t_p + \Delta) \Rightarrow (\text{true}; \lceil PF_i \rceil)) \\
\wedge \\
\Box((l \geq t_p; \lceil PF_i \rceil) \Rightarrow (\text{true}; (\lceil F_i \rceil \wedge l \geq t_p)))
\end{pmatrix}$$

Proof:

$$\begin{aligned}
& \lceil F_i \rceil \wedge l \geq t_p + \Delta \quad \{T < \frac{\Delta}{3}\} \\
\Rightarrow & \lceil F_i \rceil \wedge (l > \lceil \frac{t_p}{T} \rceil * T + 3 * T) \quad \{\text{Property 4}\} \\
\Rightarrow & (l \leq T; \lceil F_i \wedge FC_i \rceil) \wedge (l > \lceil \frac{t_p}{T} \rceil * T + 3 * T) \quad \{\text{DC}\} \\
\Rightarrow & l \leq T; (\lceil F_i \wedge FC_i \rceil \wedge (l > \lceil \frac{t_p}{T} \rceil * T + 2 * T)) \quad \{\text{Arithmetics}\} \\
\Rightarrow & \text{true}; (\lceil FC_i \rceil \wedge l \geq T_p) \quad \{\text{Property 8}\} \\
\Rightarrow & \text{true}; \lceil PF_i \rceil
\end{aligned}$$

And

$$\begin{array}{ll}
l \geq t_p; [PF_i] & \{\text{Property 9}\} \\
\Rightarrow l \geq t_p; \Diamond_l([FC_i] \wedge l \geq T_p) & \{\text{Lemma 5}\} \\
\Rightarrow l \geq t_p; \Diamond_l(\text{true}; [F_i] \wedge l \geq t_p) & \{\text{DC}\} \\
\Rightarrow \text{true}; ([F_i] \wedge l \geq t_p); \text{true} & \{\text{Property 2}\} \\
\Rightarrow \text{true}; ([F_i] \wedge l \geq t_p) &
\end{array}$$

□

Lemma 6 $\Box([Q \wedge P] \Rightarrow [D])$

Proof:

$$\begin{array}{ll}
[Q \wedge P] & \{\text{Definition of } P \text{ and } Q\} \\
\Rightarrow [Q \wedge \neg FFF_c] \vee [Q \wedge FFF_c] & \{\text{Property 7}\} \\
\Rightarrow [|y_v - y_r| \leq \epsilon] \vee [|y_v - y_r| = 0] & \{\text{Definition of } D\} \\
\Rightarrow [D] &
\end{array}$$

□

Theorem 2 $\left(\begin{array}{l} \Box([P] \wedge l \geq T_1 \Rightarrow \int \neg D < \delta * l) \\ \wedge \\ \Box((\Box([P] \wedge l \geq T_1); [\neg P]^0; [P] \Rightarrow \int \neg D < \delta * l)) \end{array} \right)$

Proof:

$$\begin{array}{ll}
[P] \wedge l \geq T_1 & \{T < \frac{\delta * T_1}{2}, \text{Property 6}\} \\
\Rightarrow [P] \wedge l \geq T_1 \wedge (l \leq T; [Q]) & \{\text{DC}\} \\
\Rightarrow (l \leq T; [Q \wedge P]) \wedge l \geq T_1 & \{\text{Lemma 6}\} \\
\Rightarrow (l \leq T; [D]) \wedge l \geq T_1 & \{\text{DC}\} \\
\Rightarrow (\int \neg D \leq T) \wedge l \geq T_1 & \{T < \frac{\delta * T_1}{2}\} \\
\Rightarrow \int \neg D < \delta * l &
\end{array}$$

And

$$\begin{array}{ll}
([P] \wedge l \geq T_1); [\neg P]^0; [P] & \{\text{P6, DC}\} \\
\Rightarrow \left((l \leq T; [Q \wedge P]; [\neg P]^0; (l \leq T \vee (l \leq T; [Q \wedge P]))) \right) & \{\text{L6}\} \\
\Rightarrow (l \leq T; [D]; l \leq T; (\Box \vee [D])) \wedge l \geq T_1 & \{\text{DC}\} \\
\Rightarrow (\int \neg D \leq 2 * T) \wedge l \geq T_1 & \{T < \frac{\delta * T_1}{2}\} \\
\Rightarrow \int \neg D < \delta * l &
\end{array}$$

□

Theorem 3 $\Box((\Box([P] \wedge l \geq T_1); \text{true} \Rightarrow \int \neg D < \delta * l))$

Proof: We need to use the induction theorem. Let

$$R(X) \stackrel{\text{def}}{=} \Box((\Box([P] \wedge l \geq T_1); [\neg P]^0; [P]; X \Rightarrow \int \neg D < \delta * l))$$

It follows from Theorem 2 that $R(\top)$ holds. We next prove

$$R(X) \Rightarrow R(X \vee (\lceil P \rceil; X) \vee (\lceil \neg P \rceil; X)).$$

Because

$$\begin{aligned} & R(X \vee (\lceil P \rceil; X) \vee (\lceil \neg P \rceil; X)) && \{\lceil \neg P \rceil = \text{false}\} \\ \Leftrightarrow & R(X \vee (\lceil P \rceil; X)) && \{\text{Definition of } R(X)\} \\ \Leftrightarrow & \Box((\lceil P \rceil \wedge l \geq T_1); \lceil \neg P \rceil^0; \lceil P \rceil; \lceil P \rceil; X \Rightarrow \int \neg D < \delta * l). \end{aligned}$$

It is sufficient to prove

$$\begin{aligned} R(X) & \Rightarrow ((\lceil P \rceil \wedge l \geq T_1); \lceil \neg P \rceil^0; \lceil P \rceil; \lceil P \rceil; X \Rightarrow \int \neg D < \delta * l) \\ & (\lceil P \rceil \wedge l \geq T_1); \lceil \neg P \rceil^0; \lceil P \rceil; \lceil P \rceil; X && \{\text{DC}\} \\ \Rightarrow & (\lceil P \rceil \wedge l \geq T_1); \lceil \neg P \rceil^0; \lceil P \rceil; (\lceil \neg P \rceil^0 \vee \lceil P \rceil^0); \lceil P \rceil; X && \{\text{DC}\} \\ \Rightarrow & \left(\begin{aligned} & ((\lceil P \rceil \wedge l \geq T_1); \lceil \neg P \rceil^0; \lceil P \rceil; \lceil \neg P \rceil^0; \lceil P \rceil; X) \\ & \vee ((\lceil P \rceil \wedge l \geq T_1); \lceil \neg P \rceil^0; \lceil P \rceil; \lceil P \rceil^0; \lceil P \rceil; X) \end{aligned} \right) && \{\text{P3, DC}\} \\ \Rightarrow & \left(\begin{aligned} & ((\lceil P \rceil \wedge l \geq T_1); \lceil \neg P \rceil^0; (\lceil P \rceil \wedge l \geq T_1); \lceil \neg P \rceil^0; \lceil P \rceil; X) \\ & \vee ((\lceil P \rceil \wedge l \geq T_1); \lceil \neg P \rceil^0; \lceil P \rceil; X) \end{aligned} \right) && \{\text{T2, } R(X)\} \\ \Rightarrow & \left(\begin{aligned} & (\int \neg D < \delta * l); \lceil \neg P \rceil^0; (\int \neg D < \delta * l) \\ & \vee \int \neg D < \delta * l \end{aligned} \right) && \{\text{DC}\} \\ \Rightarrow & \int \neg D < \delta * l. \end{aligned}$$

Therefore, $R(\text{true})$ holds, that is

$$\Box((\lceil P \rceil \wedge l \geq T_1); \lceil \neg P \rceil^0; \lceil P \rceil; \text{true} \Rightarrow \int \neg D < \delta * l).$$

It follows that

$$\begin{aligned} & (\lceil P \rceil \wedge l \geq T_1); \text{true} && \{\text{DC}\} \\ \Rightarrow & \left(\begin{aligned} & (\lceil P \rceil \wedge l \geq T_1) \\ & \vee ((\lceil P \rceil \wedge l \geq T_1); \lceil \neg P \rceil^0; \lceil P \rceil) \\ & \vee ((\lceil P \rceil \wedge l \geq T_1); \lceil \neg P \rceil^0; \lceil P \rceil; \text{true}) \end{aligned} \right) \\ \Rightarrow & \int \neg D < \delta * l. \end{aligned}$$

□

It follows from Theorems 1 and 3 that the requirements are satisfied.

7 Discussion

In the paper, we have presented a rigorous design of an FDI algorithm. Such systems are quite complex, and we have used two techniques, namely, Duration Calculus and program logic, to verify the algorithm. Our verification is not

completely formal, in that operational reasoning has been used at a number of places to incorporate the results of program verification into DC. More research is needed to provide a more formal link between DC and program logic.

We have simplified the fault model in several aspects. For example, we assume that when two sensors are faulty, their outputs will be larger than a constant G . In practice, this may not necessarily be the case, but only with a high probability. Therefore, a more precise model would include probability, and more research is needed for reasoning about such systems.

Acknowledgements We are grateful to Zhou Chaochen who has been involved in the requirement analysis and the modelling of the system. Paritosh Pandya checked some lemmas using his validity checker for DC and found a mistake.

References

1. K.R. Apt and E.-R. Olderog. *Verification of Sequential and Concurrent Programs (2nd edition)*. Springer-Verlag, 1997.
2. R.J.R. Back. *Correctness Preserving Program Refinements: Proof Theory and Applications*, volume 131 of *Mathematical Center Tracts*. Mathematical Centre, Amsterdam, 1980.
3. Zhou Chaochen and M. R. Hansen. Chopping a point. In J.-F. He, J. Cooke, and P. Wallis, editors, *Proc. BCS FACS 7th Refinement Workshop: Theory and Practice of System Design*, Bath, U.K., July 1996. Electronic Workshops in Computing, Springer-Verlag.
4. Zhou Chaochen and M. R. Hansen. An adequate first order interval logic. In H. Langmaack, A. Pnueli, and W.-P. de Roever, editors, *International Symposium, Compositionality – The Significant Difference*. Springer-Verlag, 1998, 1998.
5. Zhou Chaochen, C.A.R. Hoare, and A.P. Ravn. A calculus of durations. *Information Processing Letters*, 40(5):269–276, 1991.
6. Zhou Chaochen, A.P. Ravn, and M.R. Hansen. An extended duration calculus for hybrid systems. In *Hybrid Systems, R.L. Grossman, A. Nerode, A.P. Ravn, H. Rischel (Eds.)*, pages 36–59. LNCS 736, Springer-Verlag, 1993.
7. T.A. Henzinger, Z. Manna, and A. Pnueli. Temporal proof methodologies for timed transition systems. *Information and Computation*, 112(2):273–337, 1994.
8. Z. Manna and A. Pnueli. Verifying hybrid systems. In *Hybrid Systems, R.L. Grossman, A. Nerode, A.P. Ravn, H. Rischel (Eds.)*, pages 36–59. LNCS 736, Springer-Verlag, 1993.
9. C.C. Morgan. *Programming from Specifications*. Prentice-Hall, 1990.
10. B. Moszkowski. A temporal logic for multilevel reasoning about hardware. *IEEE Computer*, 18(2):10–19, 1985.
11. P.K. Pandya, H.-P. Wang, and Q.-W. Xu. Towards a theory of sequential hybrid programs. In D. Gries and W.-P. de Roever, editors, *Proc. IFIP Working Conference PROCOMET'98*, pages 366–384. Chapman & Hall, 1998.
12. Q.-W. Xu. Semantics and verification of extended phase transition systems in duration calculus. In O. Maler, editor, *International Workshop on Hybrid and Real-Time Systems*, pages 301–315. LNCS 1201, Springer-Verlag, 1997.

13. Q.-W. Xu and W.-D. He. Hierarchical design of a chemical concentration control system. In R. Alur, T. Henzinger, and E. Sontag, editors, *Hybrid Systems III: Verification and Control*, pages 270–281. LNCS 1066, Springer-Verlag, 1995.

Appendix

Procedure P_1^i handles the case that only sensor i is permanently faulty.

```

 $P_1^i \triangleq$ 
 $FC_j := (|x_j - x_k| \geq T_h) \wedge (|x_j - x_l| \geq T_h) \ (j, k, l \neq i);$ 
if
   $\square_{j,k,l} \neg FC_j \wedge \neg FC_k \wedge \neg FC_l \rightarrow$ 
     $count(j) := 0; count(k) := 0;$ 
     $count(l) := 0; y_v := x_k;$ 
   $\square_{j,k,l} FC_j \wedge \neg FC_k \wedge \neg FC_l \rightarrow$ 
     $count(j) := count(j) + T; count(k) := 0;$ 
     $count(l) := 0; y_v := x_k;$ 
   $\square_{j,k} FC_j \wedge FC_k \rightarrow count(j) := count(j) + T; count(k) := count(k) + T;$ 
     $count(l) := count(l) + T; y_v := FSV;$ 
fi

```

Procedure P_1^{ij} handles the case that sensor i and j are permanently faulty.

```

 $P_1^{ij} \triangleq$ 
 $FC_k := |x_k - x_l| \geq T_h \ (k, l \neq i, j);$ 
if
   $\square_k \neg FC_k \rightarrow count(k) := 0; count(l) := 0; y_v := x_k;$ 
   $\square_k FC_k \rightarrow count(k) := count(k) + T; count(l) := count(l) + T; y_v := FSV;$ 
fi

```

Scalable Data and Sensor Fusion via Multiple Agent Hybrid Systems

Wolf Kohn^{1*}, Anil Nerode^{2**}, and Jeffrey B. Remmel^{3***}

¹ Hynomics Corporation
10632 NE 37th Circle, Bldg. 23
Kirkland, WA 98003, USA
wk@hybrithms.com

² Center for Foundations of Intelligent Systems
625 Rhodes Hall, Cornell University
Ithaca, NY 14853, USA
anil@math.cornell.edu

³ Department of Mathematics, UC San Diego
9500 Gillman Drive
La Jolla, CA 92093, USA
remmel@math.ucsd.edu

Abstract. We address the problem of synchronizing estimates of plant state. The approach proposed in the study is unique because it does not attempt to transform the data to a common representation. Rather we establish a framework which we call the Multiple Agent Hybrid Estimation Architecture in which we allow heterogeneous data to flow between individual agents in the network to improve their individual estimates of the current plant state.

1 Introduction

This paper is a brief outline of our multiple agent hybrid estimation architecture (MAHEA) and its variational basis. It is designed to model fusion of heterogeneous sensor data and also to model coordination of heterogeneous sensors for distributed plants. It can be used for virtually any application in which sensor fusion of heterogeneous sensors is required. Most sensor fusion strategies may be mathematically recast as synchronization problems in the relaxed calculus of variations. The MAHEA architecture is based on our MAHCA architecture and is specifically targeted at hybrid system applications in which one has to extract digital programs for coordination and control of heterogeneous sensors. In the

* Research supported by SDIO contract DAA H-04-93-C-0113, Dept. of Commerce Agreement 70-NANB5H1164.

** Research supported by Dept. of Commerce Agreement 70-NANB5H1164 and ARO DOD MURI, "Integrated Approach to Intelligent Systems", DAAH04-96-1-0341

*** Research supported by ARO contract DAAL03-91-C-0027, SDIO contract DAA H-04-93-C-0113, Dept. of Commerce Agreement 70-NANB5H1164 and NSF grant DMS-9306427.

battlefield and for large economic and industrial systems, there is no central supervisor with complete information about the dynamics of all parts of processes in the system. MAHEA models these processes as a distributed system in which there are semi-autonomous agents, each sensing parts of certain processes. Each agent has information about the measurements made by its sensors. Each agent has only partial information about measurements made by sensors observed by other agents. This partial information is communicated by those agents via communication lines. MAHEA is an adaptation of MAHCA, our Multiple Agent Hybrid Control Architecture ([18], [16], [14], [9], [19]), to estimation problems. MAHCA itself was designed to extract digital control programs for a variety of processes including automated manufacturing, multimedia networks, controlling flexible gun tubes, flight planning for missiles, battle management, traffic management of highway systems, supply line planning and logistics, etc. MAHEA is more specifically designed for sensor fusion, distributed simulation, and coordination and synchronization of semi-autonomous units in any large business or industrial enterprise. We allow heterogeneous data to flow between individual agents in the network to improve their *individual* estimates of the current plant state based on their internal model of the dynamics of process states and of the sensors that sense the process states. We provide a brief description of a MAHEA agent, the basics of a MAHEA agent model of the plant, and the procedure it uses to improve its plant estimate over time. Each agent of MAHEA formulates a relaxed variational optimization problem whose successful resolution produces an estimate of the plant state. Each agent operates as a real-time theorem prover in the domain of relaxed variational theory. Its purpose is extraction of real time estimates of plant states and real time coordination of plants in a distributed setting with no central supervisor and incomplete information. The MAHEA architecture is based on the construction of estimation Lagrangians on a manifold designed for the problem at hand. In the dual Hamilton-Jacobi formulation, the problem is a relaxed dynamic programming problem on the manifold for computing geodesic fields. We have developed a variety of differential geometric techniques for solving such problems using connections, covariant derivatives, and Cartan derivatives associated with the Hessian form for the corresponding Lagrangian. This is not discussed here. First we give a basic mathematical model for sensor fusion. There are two fundamental problems addressed by a MAHEA agent.

Fundamental Problems.

1. Estimate the evolution of global plant state from the evolution of heterogeneous sensor readings.
2. Develop mechanisms for coordinating sensors which operate in different local clock-space coordinates so that they provide consistent estimates.

An example of 2. is the problem of synchronization of the Army's nation wide distributed simulation net. Some of the sites interacting in single theatre-wide simulations are real army-fielded units consisting of men and equipment in a real environment, some sites are purely battlefield simulators, some sites

are real personnel interacting with simulated vehicles, some sites are simulated personnel interacting with real vehicles, etc. One does not wish to shoot down a plane at one site in local time, and then at another local time and site, be sensing it as continuing to fight, due to communication delays. The Army simulators that are supposed to interact are of different generations, using software and hardware written in incompatible languages under incompatible operating systems on incompatible machines of uncertain age, with entirely different time and space granularities in the models.

2 Sensors and Measurement Maps

Sensors. Suppose the evolution of the time history of a plant is being observed by a family of heterogeneous sensors. What is observed might be the evolution of motion of a car or airplane, the evolution of the state of a battlefield, or the evolution of the state of a factory, or business. Each sensor gives sensor readings as a function of time and of the state of the plant. That is, each sensor determines a map from the space of plant states to the space of sensor readings. The observed plant always has partially unknown dynamics. One never succeeds in exact measurement of the state of an actual physical plant. All that is ever available is a lot of simultaneous readings of states of sensors.

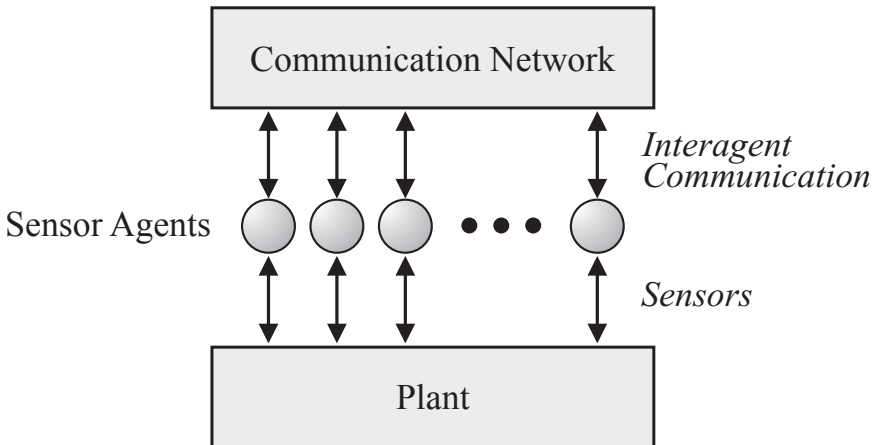


Fig. 1. Agent framework

In our multiple agent sensor estimation architecture, which is an instantiation for multiple heterogeneous sensors of our multiple agent hybrid system architecture, we equip each sensor with a software program, called a sensor estimation agent. This agent has internal to its program an executable model of sensor dynamics for the sensor(s) for which it is the agent. It also has a Lagrangian

executable model of plant dynamics. It is equipped with communication links to (some) other sensor agents. These links transmit information about the local sensors and the plant to some other sensors over a communications network. For instance, if sensors and their agents are monitoring a battlefield, their agents may be providing to other agents the results of video, radar, or infrared readings. See Figure 1.

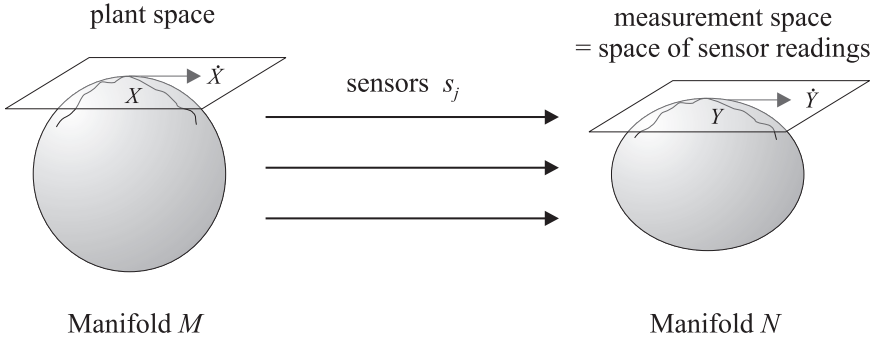
Sensors as smooth maps. Each sensor gives sensor readings as a function of time and of the state of the plant. That is, each sensor determines a map from the space of plant states to the space of sensor readings. We model the plant states as a smooth compact finite dimensional manifold M . We model each sensor as a smooth map $s : M \rightarrow \mathbb{R}$ from the manifold of plant states to the reals. Such a map is usually called a *scalar* of the manifold. If a sensor produces measurements of k quantities $s_j(x)$, we regard the sensor as defining a vector of k scalars; the sensor then maps the manifold of plant states into k -dimensional Euclidean space. We remain vague as to what “smooth” means. As is usual in applied mathematics, “smooth” means whatever is required in the immediate argument. Sometimes smooth means twice continuously differentiable; sometimes it means C^∞ ; sometimes it means analytic; whatever is required for the computation at hand.

Example. A fixed position black and white video camera transforming the observed scene into a c row d column pixel screen with intensity at a pixel as value. This defines cd scalars, one for each pixel position, the value at the pixel position being the intensity of the signal, and the vector values are in cd -dimensional Euclidean space.

Representation of Concrete Sensors. We obtain the smooth map for each sensor in a concrete problem from the differential equations governing its input output relation, equations obtained from the radar, infrared, video camera, etc. designers and manufacturers. For our MAHEA architecture the output of each sensor is treated by its agent as arising from a geodesic out of a geodesic field for a Lagrangian variational problem obtained from the differential equations for the sensor by an inverse variational method.

The variational approach to sensor fusion here differs from other approaches to sensor fusion of which we are aware in that it does not attempt to transform the *instantaneous readings* of heterogeneous sensors to an *instantaneous estimate* of plant state, but rather transforms *past evolution* of readings of heterogeneous sensors to estimate *present and future evolution* of the plant state.

The measurement map and measurement manifold. Simultaneous sensor readings by all sensors of plant state give simultaneous values of the associated scalars $s_j : M \rightarrow \mathbb{R}$ on the plant state manifold. If k is the total number of scalars, this defines a *measurement map* $\mathbf{M} : M \rightarrow \mathbb{R}^k$ given by $\mathbf{M}(x) = (s_1(x), \dots, s_k(x))$, of the plant space M onto the *measurement space* $N = \mathbf{M}(M) \subseteq \mathbb{R}^k$. We assume N is a smooth compact submanifold in $\mathbb{R}^k$. See Figure 2.



$$k \text{ sensors (scalar maps) } s_j : M \rightarrow \mathbb{R}$$

$$N = \{ (s_1(x), \dots, s_k(x)) \mid x \in M \} \subseteq \mathbb{R}^k$$

Fig. 2. Sensor fusion – single agent

For concrete sensors, we can compute the measurement map $\mathbf{M}$ from the physical equations governing the sensors as a vector of the input-output maps of the sensors. For radars, infrared sensors, video cameras, etc., these equations are standard and readily available from the designers. But they seem not to have been previously used simultaneously for plant state estimation, that is for sensor fusion, or for synchronization. The measurement map $\mathbf{M}$ tells how the plant state is transformed into a measurement vector. What we normally observe is not the trajectory of evolution of plant state on the manifold of plant states, but rather we observe the evolution of the image trajectory of measurement vectors in the measurement manifold. The measurement map concept gives a geometric meaning to sensor fusion.

The Algebra of Scalars. If we are given a compact Hausdorff space M , and $C(M)$ is the Banach algebra of continuous real valued functions on M , then by Urysohn's lemma the "measurement map" $I : M \rightarrow \mathbb{R}^{C(M)}$, given by $I(x)(f) = f(x)$ is 1-1. In the interpretation offered, $I(x)$ would represent the result of simultaneously reading the output of every possible sensor (continuous scalar) at state x . This says that if we had a sensor for every continuous scalar, we could recover the plant state x from all the sensor readings of x . Much more is true.

The Gelfand-Mazur theorem implies that when M is any compact Hausdorff space, M is naturally homeomorphic to the *spectrum* of $C(M)$, that is, to the maximal ideal space of the Banach algebra $C(M)$ ([27], p.123). There is a similar result for C^∞ manifolds M . There $C(M)$ is replaced by $C^\infty(M)$, and the spectrum of $C^\infty(M)$ carries the C^∞ -manifold structure.

Any measurement map $\mathbf{M}$ arising from a finite set of sensors (scalars) may be regarded as a finite approximation to the embedding I .

Rank n measurement map. Suppose the plant state manifold M is n dimensional. When is a set of sensors enough to recover a point x of M from its image under the measurement map $\mathbf{M}$? If in a neighborhood of a point x of M , the rank of $\mathbf{M}$ is n , then the inverse function theorem says that locally around x , we can recover the plant state trajectory γ in M from its image trajectory $\mathbf{M} \circ \gamma$ in the measurement manifold N . This is an algorithmic recovery, if the measurement map is constructive and smooth, so that the inverse function theorem can be constructively applied.

Rank $< n$ measurement map. Suppose that in the neighborhood of point x , the rank of $\mathbf{M}$ is less than the dimension n of M . How closely is the plant state trajectory at x determined by the measurement manifold trajectory through its image $\mathbf{M}(x)$? The derivative of $\mathbf{M}$ at plant state x is the Jacobian transformation mapping the tangent space at x in M to the tangent space of the measurement manifold N at $\mathbf{M}(x)$. If we think of plant state trajectory evolution as defined by the direction field of its state trajectory, then knowing the direction of the image trajectory through $\mathbf{M}(x)$ determines the direction of the pre-image plant state trajectory on M through x only up to adding any vector in the kernel of the Jacobian of $\mathbf{M}$ at x . This is the natural infinitesimal limitation on recovery of direction of a plant state trajectory through x from the image direction under the Jacobian of $\mathbf{M}$. When the kernel has dimension $d > 0$, there are d “degrees of freedom” in the determination of the plant state trajectory direction from the image measurement trajectory direction. Otherwise put, the measurement map image sees the tangent plane at x mod the kernel of the Jacobian of $\mathbf{M}$, that is, it sees an $n - d$ dimensional picture rather than the true n dimensional picture.

3 Multiple Sensors and Multiple Agents

The discussion above fits the situation where there is a single agent with perfect information about all sensor readings all the time, who can process this information instantaneously. In the battlefield and for large economic and industrial systems, there is no such central reservoir of complete information. These are distributed systems in which each agent has information about the measurements made by those sensors that that agent observes and only partial information about measurements of other sensors observed by other agents. We adapt our Multiple Agent Hybrid Control Architecture as a Multiple Agent Hybrid Estimation Architecture (MAHEA). We allow heterogeneous data to flow between individual agents in the network to improve their *individual* estimates of the current plant state based on their internal model of the dynamics of the plant state manifold. See Figure 3.

The effect of simultaneous measurements. It is important to note that simultaneous measurements by sensors each of rank $< n$ can be used to construct simultaneous measurements maps of rank n , allowing local recovery of the point x from its sensor measurements. For example, suppose $n = 3$ and we have two

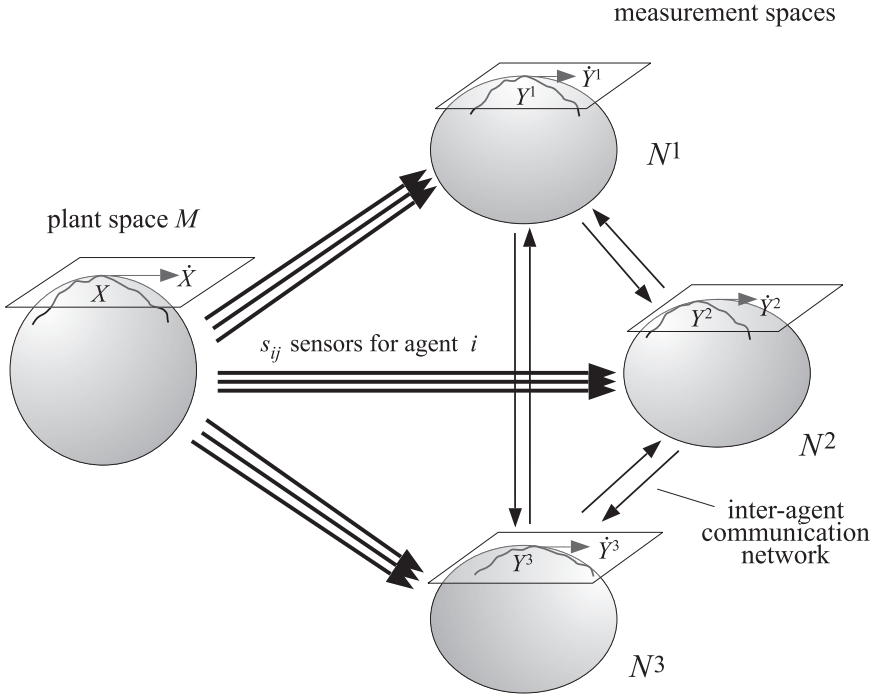


Fig. 3. Sensor fusion – multiple agents

agents $i = 1, 2$, with their measurement maps $\mathbf{M}_i : M \rightarrow \mathbb{R}^2$ on a compact manifold $M \subseteq \mathbb{R}^3$ from scalars s_{ij} for $j = 1, 2$, with each $\mathbf{M}_i$ of rank 2. Then the simultaneous measurement map $\mathbf{M} : M \rightarrow \mathbb{R}^4$ into the product space

$$\mathbf{M}(x_1, x_2, x_3) = (s_{11}(x_1, x_2, x_3), s_{12}(x_1, x_2, x_3), s_{21}(x_1, x_2, x_3), s_{22}(x_1, x_2, x_3))$$

may well have rank 3, thus allowing local recovery of the plant state (x_1, x_2, x_3) from the measurement map image $\mathbf{M}(x_1, x_2, x_3)$. This is one standard way to recover three-dimensional images from two-dimensional images from disparate sources, such as infrared, radar, or video shot from different locations.

Incorporating time. The model above needs to be supplemented for sensors which are not physically fixed relative to the observed manifold. If sensors are moving relative to a changing plant manifold, we need to introduce a time coordinate, extending the manifold, the measurement map, and the sensor equations by a time dimension.

4 Conservation Laws and Elimination Methods

In our MAHEA architecture, we use observed deviations from conservation laws about measurement map values y to correct by feedback the sensor and plant models, parameters and initial conditions. How does one compute such conservation laws? In the variational formulation, conservation laws are obtained by the Noether algorithm. In the case of sensor fusion, the analogous method based on the same principles uses symbolic calculus to eliminate x from the equations for sensors which involve input x , and output y to get differential relations on y alone, possibly infinite in number. These express the desired conservation laws. These are equations among coordinates $y = (y_1, \dots, y_m)$ of the value of the measurement map $\mathbf{M}$ which must hold at all times if the measurement values y come from at least one possible manifold state x . Any deviation from such conservation laws between the coordinates of y means that there is no possible x in the manifold M for which y is the measurement vector. That is, there is no possible state of the manifold as observed which could give rise to these measurements. This has to be traced back to incorrect or unmodelled dynamics in the description of the sensors, whether differential or variational.

There are many other elimination methods which may be of help for finding conservation laws for limited classes of problems. In case the system of differential equations in x, y from which x is to be eliminated is a system of algebraic differential equations, built up by rational operations and differentiation from variables, then Ritt's elimination procedure for differential algebra (alternately, the decision method for the theory of differentially closed fields) says that there is an algorithm which gives a set of conservation laws for the y which are necessary and sufficient for the existence of x such that (x, y) satisfy the equation. Thus in this case there is a set of conditions which can be computed and which are met by y iff the measurements arise from some plant state x . This is a kind of vanilla generalization of Noether's algorithm (and its converse since the existence of measurements y satisfying the system with x eliminated imply that the original (x, y) system is satisfiable by some plant state x .) In case the transformations are real (complex) algebraic, there are the elimination methods for real closed (algebraically closed) fields of Tarski. There are recent results on definability in o-minimal structures and hybrid systems which may prove useful if they can be improved enough to exhibit elimination of quantifiers in terms of understandable basic predicates.

5 Noether Synchronization of Agents

Geodesic Fields. We employ Caratheodory's definition ([5], [25], p.188). On a bounded region $X \subseteq M$ of an n -dimensional manifold M , a *geodesic field* for a Lagrangian L is defined by:

1. a set of n C^2 -functions $\phi_j(t, x)$ defined on the product $P = [\tau_0, \tau_1] \times X$ of a finite closed time interval and the region X , such that the integral curves of

- $\dot{x} = \phi_j(t, x)$, for $j = 1, \dots, n$, yield a family of curves which cover P simply; and
2. a C^2 -function $S(t, x)$ with the same domain P such that if a new Lagrangian is defined by

$$L^*(t, x, \dot{x}) = L(t, x, \dot{x}) - \frac{\partial S}{\partial t} - \frac{\partial S}{\partial x_j} \dot{x}_j \quad (1)$$

(which obviously has the same geodesics as L in the region) then $L^*(t, x, \dot{x}) = 0$ whenever $\dot{x} = \phi(t, x)$, and $L^*(t, x, \dot{x}) > 0$ otherwise.

The existence of a geodesic field and $\det \left(\frac{\partial^2 L}{\partial \dot{x} \partial \dot{x}} \right) \neq 0$ implies that the Weierstrass excess function is non-negative:

$$E(t, x, \phi, \dot{x}) = L(t, x, \dot{x}) - L(t, x, \phi) - \frac{\partial L(t, x, \phi)}{\partial \dot{x}_j} (\dot{x}_j - \phi_j) \geq 0 \quad (2)$$

Then a corresponding S can be computed from the Hamilton-Jacobi equation $\frac{\partial S}{\partial t} + H(t, x, \frac{\partial S}{\partial x}) = 0$ where $p = \frac{\partial L}{\partial \dot{x}}$, and $\det \left(\frac{\partial^2 L}{\partial \dot{x} \partial \dot{x}} \right) \neq 0$ implies $\dot{x} = \phi(t, x, p)$, and the Hamiltonian is defined by

$$H(t, x, p) = -L((t, x, \phi(t, x, p)) + p\phi(t, x, p) \quad (3)$$

(see [25], pp.194-198.)

We assume that each agent is an estimation agent, estimating the current state of a manifold x at time t . This may be in applications a measurement manifold rather than the original plant state manifold. In accordance with the general MAHCA approach we assume that the model of state evolution is expressed so that evolution of state is a geodesic – i.e. satisfies the Euler-Lagrange equations – corresponding to a non-negative Lagrangian cost function. Each agent has a Lagrangian, and is estimating state evolution as a geodesic according to his or her own Lagrangian. The agents, whose sensors observe the same manifold, may have quite different Lagrangians. (Think of two radars observing the same field, or of a radar and an infrared video camera observing the same field. Each gives a current pixel image estimate. What is the relation between the images?) We will assume that each agent has a Lagrangian with a given geodesic field in the sense of Caratheodory, say over the same region, but based on different Lagrangians. What definition can we give for synchronizing their states? Suppose that the local time and state for agent 1 is (t_1, x_1) (state-clock coordinates). Define a (local) *synchronization of state-clock coordinates* of agent 1 with those of agent 2 as a smooth function $\Psi(x, t)$ such that in a neighborhood of (t_1, x_1) , we have

$$L_1(t, x, \dot{x}) = L_2(\Psi_t(x, t), \Psi_x(x, t), \dot{\Psi}_x(x, t)) \quad (4)$$

This means that one side is zero if and only if the other is zero, and according to the definition of geodesic field, one side is greater than zero if and only if the

other is. Thus in the domain involved, $(x, \dot{x}, t)$ is on the geodesic field for agent 1 if and only if $(\Psi_t(x, t), \Psi_x(x, t), \dot{\Psi}_x(x, t))$ is on the geodesic field for agent 2. The meaning of $\Psi(x, t)$ is that if agent 2 knows Ψ and her own state-clock coordinates, she knows corresponding state-clock coordinates for agent 1 in the sense that costs agree along corresponding geodesics. We are assuming that cost plus being on the manifold is all that counts. Similarly if agent 2 knows a synchronization Φ , such that

$$L_2(t, x, \dot{x}) = L_1(\Phi_t(x, t), \Phi_x(x, t), \dot{\Phi}_x(x, t)) \quad (5)$$

in a neighborhood of local state and time, he knows the corresponding time, state and direction that gives agent 1 the same costs along her geodesics. So for our form of synchronization and coordination, we need to know how to compute local synchronizations. This is where the Noether algorithm comes in.

To illustrate how this method is used, suppose that there is a global Lagrangian state estimation function for the plant $L(t, x, \dot{x})$ and we have two agents, Agent 1 with its estimation Lagrangian $L_1(t_1, x_1, \dot{x}_1)$ and Agent 2 with its estimation Lagrangian $L_2(t_2, x_2, \dot{x}_2)$. Moreover assume (in order to get implicit equations for them) that we have state and clock transition functions for Agent $i = 1, 2$, given by:

$$\begin{aligned} x_i^j &= x_i^j(t, x, w), & j &= 1, \dots, n \\ t_i &= t_i(t, x, w) \end{aligned} \quad (6)$$

where $w = (w^1, \dots, w^n)$ is a set of auxiliary parameters. Embedding in a smooth family with a auxiliary parameter w is the basis of Lie's and Noether's method of finding symmetries for differential equations and variational problems. The w^i have a simple meaning in the sensor interpretation. They are the sensor readings. Thus $\frac{\partial x_i^j}{\partial w^k}$ is the rate of change, or sensitivity, of state variable x_i with respect to the k -th sensor reading. We want to determine $x_i^j(t, x, w), t_i(t, x, w)$, solving

$$L_i(x_i(t, x, w), x_i(t, x, w), \dot{x}_i(t, x, w)) = L(t, x, \dot{x}) \quad (7)$$

for $i = 1, 2$.

Next we state the Noether Invariance relations. Define two classes of infinitesimal transformations for each Agent i :

$$\left. \frac{\partial x_i^j}{\partial w^k} \right|_{w=0} = \theta_{i,k}^j(t_i, x) \quad (8)$$

and

$$\left. \frac{\partial t_i}{\partial w^k} \right|_{w=0} = \tau \quad (9)$$

Next let $E_{i,j}$, for $i, j = 1, \dots, n$, be the Euler-Lagrange operators for L_i . (see [25], p.185) That is, let

$$E_{i,j}(L_i) = \frac{d}{dt_i} \left(\frac{\partial L}{\partial \dot{x}} \right) - \frac{\partial L_i}{\partial x_i^j} \quad (10)$$

This given, the Noether Invariance Relations (see [25], pp.201-207) are given by

$$\sum_{j=1}^n E_{i,j}(L_i)(\theta_{i,k}^j - \dot{x}_i^j \tau_{i,k}) = \frac{d}{dt} \left(L_i \tau_{i,k} + \sum_{j=1}^{n_i} \frac{\partial L_i^j}{\partial \dot{x}_i^j} (\theta_{i,k}^j - \dot{x}_i^j \tau_{i,k}) \right) \quad (11)$$

for $k = 1, \dots, n$

If the estimation Lagrangian L_i has an extremal, i.e. if there exists a curve $x_i(t) = \gamma_i(t)$ which over the usual admissible family of curves minimizes

$$\int_{t_0}^{t_0+\Delta} L_i(t_i, x_i, \dot{x}_i) dt_i \quad (12)$$

where $\gamma_i(t_0) = x_0$ and $\gamma_i(t_0 + \Delta) = x_1$, then L_i must satisfy the Euler-Lagrange equations $E_{i,j}(L_i) = 0$ for $j = 1, \dots, n_i$. In that case the left hand sides of (11) are 0. Hence there are constants $c_{i,k}$ such that

$$L_i \tau_{i,k} + \sum_{j=1}^{n_i} \frac{\partial L_i^j}{\partial \dot{x}_i^j} (\theta_{i,k}^j - \dot{x}_i^j \tau_{i,k}) = c_{i,k} \quad (13)$$

for $k = 1, \dots, n$. These $c_{i,k}$ are determined from the initial conditions. First, note that, given the constants $c_{i,k}$, we can solve for the infinitesimals $\tau_{i,k}$ and $\theta_{i,k}^j$ and integrate to recover the desired transformation x_i^j and t_i . We will illustrate this calculation with a simple example below. Second, note that we can monitor the failure of synchronization of the state estimation Lagrangians by simply observing that the left hand side of a Noether invariance relation is not a constant. If the left hand side of (13) is not a constant, then we know that the current agent Lagrangian is not compatible with the system Lagrangian so that Agent i would use our Adapter (see below) to reset his Estimation Lagrangian.

6 Simplified Radar Returns Example

This example is to show how easy the infinitesimal calculations can be. (See [4] section 2.2, [26], section 2.1.) A simplified range model of radar returns has the following system Lagrangian:

$$L = t^2 \left(\frac{\dot{x}^2}{2} - \frac{x^6}{6} \right) \quad (14)$$

Now suppose that Agent 1 has infinitesimal transformations

$$t_1 = t + \tau(x, t)w \quad (15)$$

$$x_1 = x + \theta(x, t)w. \quad (16)$$

Here w represents the radar returns, which are the sensor outputs. We assume that $x = x(t)$ satisfies the Euler-Lagrange equation so that

$$\begin{aligned} 0 &= \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{x}} \right) - \frac{\partial L}{\partial x} \\ &= 2t\dot{x} + t^2\ddot{x} - t^2x^5 \end{aligned} \quad (17)$$

Thus

$$t^2\ddot{x} = t^2x^5 - 2t\dot{x} \quad (18)$$

Then from (11), we get

$$\frac{d}{dt} [L\tau + \frac{\partial L}{\partial \dot{x}}(\theta - \dot{x}\tau)] = \frac{d}{dt} [t^2\dot{x}\theta - (\frac{t^2\dot{x}^2}{2} + \frac{t^2x^6}{6})\tau] = 0 \quad (19)$$

Expanding the derivative, replacing $t^2\ddot{x}$, and collecting terms, we obtain the following relations:

$$\dot{x}^0 : \quad -\frac{tx^6}{3}\tau + t^2x^5\theta - \frac{t^2x^6}{6}\frac{\partial\tau}{\partial t} = 0 \quad (20)$$

$$\dot{x}^1 : \quad t^2\frac{\partial\theta}{\partial t} - \frac{t^2x^6}{6}\frac{x^6}{6}\frac{\partial\tau}{\partial x} = 0 \quad (21)$$

$$\dot{x}^2 : \quad t\tau + t^2\frac{\partial\theta}{\partial x} - \frac{t^2}{2}\frac{\partial\tau}{\partial t} = 0 \quad (22)$$

$$\dot{x}^3 : \quad -\frac{t^2}{2}\frac{\partial\tau}{\partial x} = 0 \quad (23)$$

Note that by (23), $\tau = \tau(t)$ is just a function of t . Also (23) and (21) imply that $\frac{\partial\theta}{\partial t} = 0$ so that $\theta = \theta(x)$ is just a function of x . This means that (19) and (22) are just ordinary differential equations. Solve, getting:

$$\tau = at \quad \text{and} \quad \theta = -\frac{a}{2}x \quad (24)$$

where a is an arbitrary constant. Hence

$$t_1 = t + atw \quad \text{and} \quad x_1 = x - \frac{a}{2}xw \quad (25)$$

The constant a is determined by initial conditions.

7 Inverse Lagrangians

The methodology behind MAHCA replaces differential descriptions of evolution of state by corresponding descriptions of evolution of state along geodesics of a variational problem. This is instantiated in MAHEA in that differential models of agent sensors are replaced by Lagrangian cost function models whose geodesics generate the same state estimates. Such cost functions carry information not contained in their the Euler-Lagrange equations about connections (gauge potentials) and curvature (field strength), as in gauge field theory. (For an introduction to these topics, see[8], Ch. 10. For significant physical applications, see [6], pp.79-173.) Different inverse Lagrangians may have different properties when used for coordination and synchronization as well, and we too wish to make heavy use of associated connections and curvature [19].

Here is a brief outline of one inverse method. Interpret the differential equation as a description of a sensor as an input-output device. Suppose that $\dot{x} = g(t, x, u(t))$ describes a sensor which at time t with input reading $u(t)$ outputs measurement $x(t)$.

Define $f(t, x) = g(t, x, u(t))$, so that the equation is now $\dot{x} = f(t, x)$. Differentiate with respect to time and now write the result as $\ddot{x} = F(t, x, \dot{x})$. In the second order Euler-Lagrange equations for a to-be-determined Lagrangian $L(t, x, \dot{x})$, replace $\ddot{x}$ by $F(t, x, \dot{x})$. Differentiate the resulting equation with respect to t , and get a vector differential equation

$$\frac{d}{dt}L_{\dot{x}\dot{x}} + L_{\dot{x}x} + L_{\dot{x}\dot{x}}F_{\dot{x}} - L_{\dot{x}x} = 0 \quad (26)$$

with transpose

$$\frac{d}{dt}L_{\dot{x}\dot{x}} + L_{x\dot{x}} + F_{\dot{x}}^T L_{\dot{x}\dot{x}} - L_{\dot{x}x} = 0 \quad (27)$$

Add these two equations, divide the result by two, and get a Lyapunov equation for $\Psi = L_{\dot{x}\dot{x}}$:

$$-\frac{d}{dt}\Psi = \frac{1}{2}F_{\dot{x}}^T\Psi + \frac{1}{2}\Psi F_{\dot{x}} \quad (28)$$

Solve this equation for $\Psi = L_{\dot{x}\dot{x}}$. Then integrate along the integral curves of the original differential equation to compute L .

Remark. If we follow Weierstrass and add time as a state coordinate ([29], Ch 2, sec 7) so that $y = (x_1, \dots, x_n, t)$ and $L^*(y, \dot{y}) = L(t, x, \dot{x})$, we get what is called a *homogeneous* Lagrangian: $L^*(y, \dot{y})$, that is one such that:

$$\bar{L}^*(y, \lambda \dot{y}) = \lambda L^*(y, \dot{y}) \quad (29)$$

for all positive λ , for which the extremals are essentially the same as those of L . Differentiation of this equation with respect to λ gives Euler's theorem:

$$L^*(y, \dot{y}) = \Sigma L_{\dot{y}_i}^*(y, \dot{y}) \dot{y}_i \quad (30)$$

([28], p.4, 1.11). A little more computation shows that $(L^*)^2(y, \dot{y}) = \frac{1}{2}\dot{y}^T(L^2)_{\dot{x}\dot{x}}\dot{x}$, ([28], p.6, 1.19b). This eliminates some integration in finding an L^* and therefore an L . This is the origin of our Finsler space approach to hybrid systems ([19]).

8 The Multiple Agent Hybrid Estimation Architecture

In this section, we describe the main operational and functional characteristics an agent in a MAHEA network. We take for granted the reader has read an account of the MAHCA (Multiple Agent Hybrid System architecture) such as [18], [12]. Unexplained notions are from these papers. The MAHEA Architecture is implemented as a distributed system composed of agents and a communication network which we call the logic communication network. The architecture realizing this system operates as an on-line distributed theorem prover. Our architecture interacts with the plant at a series of update times $\Delta_1 < \Delta_2 < \dots$. These update times are a function of the application. The next update time is determined by the interaction between the plant and the agents. At an update time, each active agent will receive information from a certain suite of sensors and generate estimation actions as a side effect of proving an existentially quantified lemma which encodes the model of the plant as viewed by the agent. The conjunction of lemmas at each instant of time, encodes the desired behavior of the entire network. The number of agents in the network is variable. That is, the system can spawn new agents and deactivate agents as a function of system demand. Each agent of MAHEA, as a specialization of MAHCA, consists of five modules: a Planner, a Dynamic Knowledge Base, a Deductive Inferencer, an Adapter and a Knowledge Decoder. See Figure 4. We briefly review the functionality of an agent in terms of its modules.

The basic architecture of an estimation agent consists of five modules with the following functionality:

1. **Planner.** The Planner constructs and repairs the agent state estimation optimization criteria which we refer to as the Estimation Lagrangian associated with the agent. In particular, the Planner generates a statement representing the desired model of the estimation system as an existentially quantified logic expression herein referred to as the Estimation Statement.
2. **Inferencer** The Inferencer determines whether there is a state estimate for the agent's relaxed variational state estimation problem which is a near optimal solution where the agent's Estimation Lagrangian is used as a cost function. If there is such a solution, the agent infers a near optimal estimation and sends data to the other agents. Otherwise it infers failure terms and a new state for the agent and reports the failure to the other agents. In particular, the Inferencer determines whether the Estimation Statement is a theorem in the theory currently active in the knowledge base. If the Estimation Statement logically follows from the current status of the Knowledge Base, the inferencer generates, as a side effect of proving this Estimation Statement to be true, the current state estimate of the plant. If the Estimation

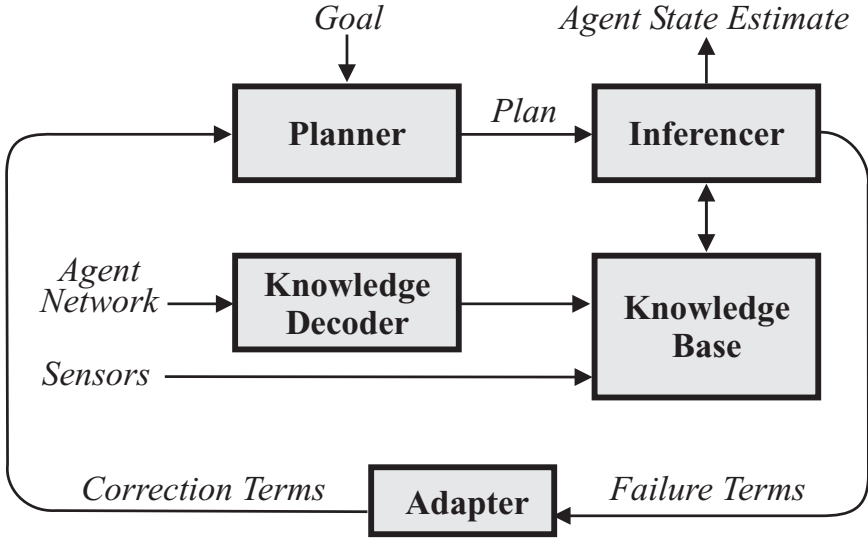


Fig. 4. MAHEA agent architecture

Statement does not logically follow from the current status of the Knowledge Base, that is, the desired behavior is not realizable, the inferencer transmits the failed terms to the Adapter module for replacement or modification. The term “near optimal” is used deliberately, since, as in the MAHCA architecture, relaxed solutions to the variational problems are used, which cannot generally be physically realized, but arbitrarily close approximations to them can be computed and realized.

3. **Adapter** The Adapter repairs failure terms and constructs correction terms.
4. **Knowledge Base** The Knowledge Base stores and updates the agent’s plant model and constraints. The Knowledge Base also stores the requirements of operations or processes within the scope of the agent’s estimation problem. It also encodes system constraints, inter-agent protocols and constraints, sensory data, operational and logic principles and a set of primitive inference operations defined in the domain of equational terms.
5. **Knowledge Decoder** The Knowledge Decoder receives and translates the other agent’s data.

To better understand how these five modules of the MAHCA architecture function in the MAHEA architecture, we outline the basic elements of an agent’s model and how it behaves. Let A_i , $i = 1, \dots, N(t)$ denote the agents active at the current time t . In our model, t takes values on the real line $\mathbb{R}$. At each time t , the status of each agent in the network is given by a point in a smooth manifold M . The Estimation Lagrangian L_i of an active agent A_i is given by a continuous function

$$L_i : M \times \mathbb{R} \rightarrow \mathbb{R}^+$$

where $\mathbb{R}^+$ is the positive real line. A point p in the manifold M is represented by a data structure of the form:

$$p(id, proc(proc\_data), est(estimation\_data), in(synch\_data), mp(mult, data))$$

Here id is an identifier taking values in a finite set ID , $proc()$ is a relation characterizing the status of plant processes, which depends on a list of parameters labeled $proc\_data$ which define the operational, load, and timing characteristics of the process involved. The relation $est()$ captures attributes of the plant being represented which depends on a list of parameters labeled $estimation\_data$ which characterize, among other things, various constraints of the plant representation of an agent at a level of abstraction compatible with the logic communication network. The relation $in()$ carries synchronization information of the logic communication network. This includes information such as priority level, connectivity and time constants. Finally, the relation $mp()$ carries multiplicity information, that is, it represents the level of network usability at this point. The associated parameter list, $mult\_data$, is composed of statistical parameters reflecting the logic network's load.

From an agent's point of view, the dynamics of the plant is characterized by certain trajectories on the manifold M . These trajectories represent the agent estimate of the state of the plant plus the flow of information through the network and its status. Specifically, we need to define two items:

(i) The Estimation Lagrangian functions :

$$\{L_i(p, t) : i \in I(t)\}$$

where $I(t)$ is the set of active agents at time t and

(ii) the actions or estimates issued by the agents.

These actions are implemented as derivations on the algebra of scalars of the manifold. The general structure of an Estimation Lagrangian function for an active agent i at time t is given by:

$$L_i(p, t) = F_i(U_i, L, \alpha_i)(p, t)$$

where F_i is a smooth function, L is the vector of Estimation Lagrangian functions, U_i is the state estimation error function, and α_i is the command action issued by the i -th agent.

In our application, each command issued by the MAHEA agent is implemented as a vector field on the manifold M . Each agent constructs its command field as a combination of "primitive" predefined vector fields. An integral curve associated with a vector field v , denoted by $\Psi(t, p)$ is termed the *flow* generated by v if it satisfies the following conditions:

$$\begin{array}{ll} \text{semigroup:} & \Psi(t, \Psi(\tau, p)) = \Psi(t + \tau, p) \\ \text{initial condition:} & \Psi(0, p) = p \\ \text{flow:} & \frac{d}{dt}\Psi(t, p) = v|_{\Psi(t, p)} \end{array}$$

Let us customize these MAHCA notions for the MAHEA model. An agent A_i is active. Let $\Delta > 0$ be the width of the current decision interval, $[t, t + \Delta)$. Let $U_i(p, t)$ be the state estimation error at the beginning of the interval. Agent A_i has a set of primitive actions:

$$\{ v_{i,j} \mid j = 1, \dots, n_i, \text{ where } v_{i,j}|_p \in TM_p \text{ for each } p \in M \}$$

where TM_p is the tangent manifold at p , and as is usual, in geometric control, control actions and vector fields are identified.

During the interval $[t, t + \Delta)$, agent A_i schedules one or more of these actions to produce a flow which will reduce the state estimation error. As in MAHCA, the schedules are determined by computing an approximate solution to a relaxed variational problem. In particular, A_i determines the fraction $\alpha_{i,j}(p, t)$ of Δ that action $v_{i,j}$ must be executed as a function of the current estimation requests $S_{r,i}(t, p)$ and the vector of estimation Lagrangians $L(p, t) = (L_1(p, t), \dots, L_{N(t)}(p, t))$ of the active agents in the MAHEA network. We can express the change in the state estimation error U_i due to the flow over the interval Δ in terms of $v_i|_p$. The evolution of the state estimation error U_i over the interval starting at point p and ending at a point p'' is given by

$$U_i(t + \Delta, p'') = U_i(t, \Psi_i(t + \Delta, p)) \quad (31)$$

At time t and at point $p \in M$ the estimation error function of agent i is given by :

$$U_i(p, t) = U_i(p, t^-) + S_{r,i}(p, t) + \sum_k Q_{i,k} L_k(p, t^-) \quad (32)$$

where t^- is the end point of the previous update interval, $S_{r,i}$ is the estimation request function to agent i , and $Q_{i,k}$ is a multiplier determining the required degree of accuracy and the urgency of the estimate of the Agent k that Agent i requires. This allocation is determined from the characteristics of the process both agents are estimating and from the process description encoded in the agent's knowledge base. The actual request from agent k to agent i is thus the term, $Q_{i,k} L_k(p, t^-)$. The information sent to agent i by agent k is the state estimation function $L_k(p, t^-)$ at the end of the previous interval. Finally the point $p \in M$ carries the current estimate of the process monitored by the agents. Agent k thus contribute to Agent i 's new estimate only if $Q_{i,k} \neq 0$.

This concludes our description of the model. The strategy for activation and deactivation of agents is omitted.

9 Last Remarks

We believe that the MAHEA agent network is an efficient mechanism for state estimation from sensor readings which is extensible, robust, scalable, allows cross-checking, and support heterogeneous sensing of heterogeneous processes.

Deployment of an agent-based system is very simple. As soon as a new source of information is available, a new estimation agent is spawned whose Knowledge Base is a model of the plant covered by that source (extensibility). Moreover no common representation of the data is required, so the system supports heterogeneous information sources. Thus our architecture allows us to incorporate existing models and estimation techniques. The built-in invariance (conservation) conditions provide tests for the validity of the data (cross-checking).

The existing theory of MAHCA (multiple agent hybrid control architecture), based on approximations of relaxed variational calculus geodesics (which is all that is available in non-convex problems) is adapted to the state estimation problem for the plant by constructing a Lagrangian which becomes 0 at points which correspond to consistent estimates of the plant and is positive at points which are not consistent. This property ensures that when an agent reaches consistent estimates of the plant, the evolution of the system produced by the flow of the corresponding geodesic field is adapted to the current information of the plant as viewed by each agent.

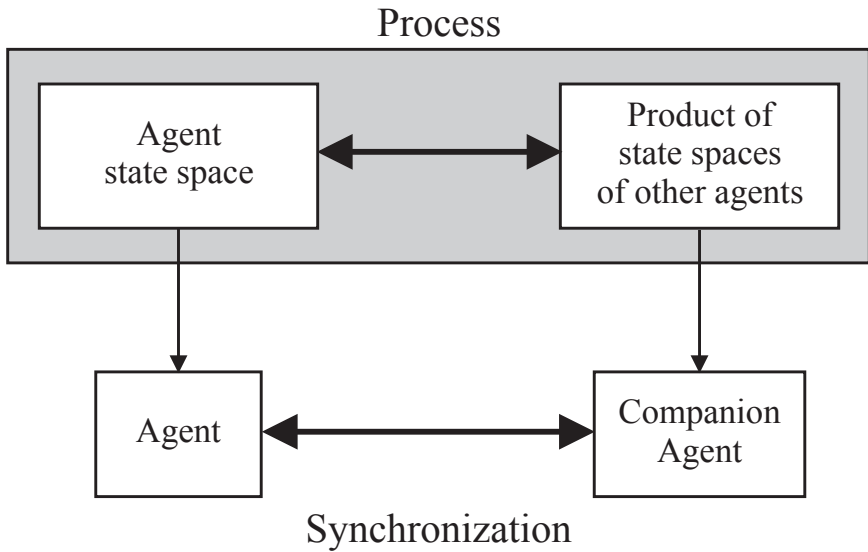


Fig. 5. The companion agent

A key result for agent synchronization is the analog of the Thevenin Theorem which states in a network with many agents, an individual agent A can view the rest of the sensor agents as a single aggregated estimation agent $C(A)$ called A 's companion agent. See Figure 5. The fact that an agent sees the rest of the estimate agent network as a single equivalent estimation agent is the basis for easy scaling up to more agents.

The robustness of the sensor agent's estimates follows from the continuity of the relaxed variational geodesic in the parameters of the problem.

The techniques mentioned in this paper for synchronization and consistency of state estimates solve many synchronization problems for our Multiple Agent Hybrid Control Architecture. A full exposition will be given in a book in preparation.

Acknowledgments. We thank Prof. John Hubbard for valuable remarks; Jennifer Davoren for work on the manuscript, and Xi Krump for the illustrations.

References

1. Alur, R., Henzinger, T. A., Sontag, E. D., eds. *Hybrid Systems III*, Lecture Notes in Computer Science **1066** (Springer-Verlag, Berlin, 1996).
2. Antsaklis, P., Kohn, W., Nerode, A., and Sastry, S. eds., *Hybrid Systems II*, Lecture Notes in Computer Science **999** (Springer-Verlag, Berlin, 1995).
3. Antsaklis, P., Kohn, W., Nerode, A., and Sastry, S. eds., *Hybrid Systems IV*, Lecture Notes in Computer Science **1273** (Springer-Verlag, Berlin, 1997).
4. Bluman, G. K., Kumei, S., *Symmetries and Differential Equations* (Springer-Verlag, Berlin, 1989).
5. Caratheodory, C., *Calculus of Variations and Partial Differential Equations of the First Order* (Dover reprint, 1982)
6. Atiyah, M., *Collected Works, vol. 5* (Oxford Science Publications, Clarendon Press, 1987)
7. Crossley, J.N., Remmel, J.B., Shore, R.A. and Sweedler, M.E., *Logical Methods: In Honor of Anil Nerode's Sixtieth Birthday* (Birkhauser, Boston, 1993).
8. Darling, R.W.R., *Differential Forms and Connections* (Cambridge University Press, 1994)
9. Ge, X. , W. Kohn, A. Nerode, and J.B. Remmel, "Multiple agent hybrid control: carrier manifolds and chattering approximations to optimal control," *Proceedings of the 33rd IEEE Conference on Decision and Control CDC'94* (IEEE Press, 1994), 4221-4227.
10. Ge, X., Kohn, W., Nerode, A. and Remmel, J.B., "Hybrid Systems: Chattering Approximations to Relaxed Controls", in [1] (1996); 76-100.
11. Gelfand, I.M., and Fomin, S.V., *Calculus of Variations* (Prentice Hall, 1963).
12. Grossman, R.L., Nerode, A., Ravn, A., Rischel, H. eds., *Hybrid Systems*, Lecture Notes in Computer Science **736** (Springer-Verlag, Berlin, 1993).
13. Kohn, W., "Declarative Control Architecture" *Communications of the ACM*, Aug 1991, Vol. 34, No. 8.
14. Kohn W., Nerode, A., "Multiple-Agent Hybrid Systems", *Proceedings of the 31st IEEE Conference on Decision and Control CDC'92* (IEEE Press, 1992); 2956-2972.
15. Kohn, W., Nerode, A., "An Autonomous Systems Control Theory: An Overview", *Proceedings of IEEE CACSD'92*, March 17-19, Napa, Ca.; 200- 220.
16. Kohn, W., Nerode, A. "Multiple Agent Hybrid Control Architecture", in [12] (1993); 297-315.
17. Kohn, W., Nerode, A., "Models For Hybrid Systems: Automata, Topologies, Controllability, Observability", in [12] (1993); 317-356.

18. Kohn W., Nerode, A., "Multiple Agent Hybrid Control Architecture", in [7] (1993); 593-623.
19. Kohn, W., Nerode, A., Rummel, J.B., "Hybrid Systems as Finsler Manifolds: Finite State Control as Approximation to Connections", in [2] (1995); 294-321.
20. Kohn, W., Nerode, A., Rummel, J. B. , Yakhnis, A., "Viability in hybrid systems," *Theoretical Computer Science* **138** (1995), 141-168
21. Kohn, W., Nerode, A., Rummel, J.B., "Feedback Derivations: Near Optimal Controls for Hybrid Systems", *Proceedings of 1996 IMACS Multiconference on Computational Engineering in Systems Applications (CESA'96)*, vol. 2, (1996), 517-521.
22. Kohn, W., Nerode, A., Rummel, J.B., "Continualization: A Hybrid Systems Control Technique for Computing", *Proceedings of 1996 IMACS Multiconference on Computational Engineering in Systems Applications (CESA'96)*, vol. 2, (1996), 507-511.
23. Kohn, W., Nerode, A., Rummel, J.B., "Agent Based Velocity Control of Highway Systems," in [3] (1997); 175-214.
24. Kohn, W., Rummel, J. B. , Nerode, A. , "Automaton Comparison Procedure for the Verification of Hybrid Systems," *Proceedings of 5th IEEE Mediterranean Conference on Control and Systems (MED'97)*.
25. Lovelock, D., and Rund, H., *Tensors, Differential Forms, and Variational Principles* (Dover, New York, 1989).
26. Olver, P. J., *Applications of Lie Groups to Differential Equations, Second Edition* (Springer-Verlag, Berlin, 1993).
27. Rickart, C., *General Theory of Banach Algebras* (D. Van Nostrand, 1960).
28. Rund, H. *The Differential Geometry of Finsler spaces* (Springer-Verlag, 1959).
29. Rund, H., *The Hamilton-Jacobi Theory in the Calculus of Variations* (D. van Nostrand, 1966).
30. Sternberg, S., *Differential Geometry* (Dover, New York, 1983).
31. Warga, K., *Optimal Control of Differential and Functional Equations* (Academic Press, New York, 1977).
32. Young, L.C., *Optimal Control Theory* (Chelsea Publishing Co., New York, 1980).

Hybrid Control Systems Using Timed Petri Nets: Supervisory Control Design Based on Invariant Properties*

Xenofon D. Koutsoukos** and Panos J. Antsaklis

Department of Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556
`xkoutsou,antsaklis.1@nd.edu`

Abstract. In this paper, a class of timed Petri nets named programmable timed Petri nets is used for supervisory control of hybrid systems. In particular, the transfer of the continuous state to a region of the state space under safety specifications on the discrete and continuous dynamics is addressed. The switching policy is embedded in the dynamics of the underlying Petri net structure and the supervisors are described by Petri nets. The discrete specifications are expressed in terms of linear constraints on the marking vector and are satisfied by applying supervisory control of Petri nets based on place invariants. The hybrid system switches from a subsystem to another, in a way that the state gradually progresses from one equilibrium to another towards the desired target equilibrium. The supervisory control algorithm is designed to allow switchings to occur only on the intersection of the invariant manifolds. Finally, in the case when the continuous dynamics are described by first order integrators, the design algorithm is formulated as a linear programming problem.

1 Introduction

In hybrid systems the behavior of interest is governed by interacting continuous and discrete dynamic processes. Hybrid control systems typically arise from the interaction of discrete planning algorithms and continuous processes, and their study is essential in designing discrete event supervisory controllers for continuous systems, and central in designing intelligent control systems with a high degree of autonomy. The investigation of hybrid systems is creating a new and fascinating discipline bridging control engineering, mathematics and computer science; further information on hybrid systems may be found in references [1,2,3,4,5,6]; see also the survey paper [7].

* This work was supported in part by the National Science Foundation grant ECS95-31485.

** This author gratefully acknowledges the fellowship of the Center of Applied Mathematics at the University of Notre Dame for the academic year 1997-98.

This paper considers systems that arise when computers are used to supervise or synchronize the actions of subsystems described by continuous dynamics (that involve continuous variables). Examples of such systems arise in chemical process control, command and control networks, power distribution networks, as well as distributed manufacturing systems. The size and complexity of such systems often requires that the system use a number of distinct operational modes. Consequently, these systems can be viewed as *supervised* systems, in which a high-level discrete (-event) supervisor is used to coordinate the actions of various subsystems so that overall system safety is guaranteed. By *safety*, we mean that pre-specified limits or tolerances on the subsystem states are not violated. In the paper, the sets of safe states are characterized by Lyapunov functionals and are related to stability properties of the subsystems. So, these systems can be viewed as a *hybrid* mixture of systems with continuous dynamics (continuous variables) supervised by a switching law generated by a (discrete-event) supervisor described by discrete dynamics (discrete variables).

Petri nets have been used extensively as a tool for modeling, analysis and synthesis for discrete event systems. For DES control, Petri nets modeling formalism offers some advantages over finite automata, and it is also useful for hybrid systems control. Peleties and DeCarlo [8] presented a model based on the work in [9] on the periodicity of symbolic observations of piecewise smooth discrete-time systems. This hybrid model is suitable for Petri net based symbolic analysis of hybrid systems; the continuous plant is approximated by a Petri net and a supervisor consisting of two communicating Petri nets controls the behavior of the open plant. Lunze *et al.* [10] proposed a model where Petri nets are used as a discrete event representation of the continuous variable system; the system and the interface are represented by a Petri net and the supervisor represents a mapping of the output event sequence into the input event sequence. Several other approaches to modeling of hybrid systems that use Petri nets have also been reported in the literature [11,12,13,14,15,16].

In this paper, a class of timed Petri nets named *programmable timed Petri nets* [17] is used to model hybrid systems. In particular, it is assumed that the switching policy is embedded in an underlying Petri net structure and that the supervisors are described also by Petri nets. Petri nets are used instead of finite automata because of the following two reasons. The first is the expressiveness of Petri nets. Petri net languages include regular languages described by finite automata and further, they can model switching policies that describe conflict, concurrency, synchronization, and buffer sizes. The second reason is that recent results in the supervisory control of discrete-event systems using ordinary Petri nets [18] have made possible to design supervisors in an efficient and transparent manner; and this methodology is used in this paper.

In the nonlinear control literature, switching has been used to expand the domain of attraction of a control system [19,20]. Here, it is assumed that the continuous subsystems admit a family of equilibria and each equilibrium has a domain of attraction associated with it. The hybrid system switches from a subsystem to another, in a way that the state gradually progresses from one

equilibrium to another towards the desired target equilibrium. For the hybrid systems of interest in this paper, this idea can be formalized using an *invariant based approach to the design of hybrid systems* [21,22]. This approach introduces the notion of a *common flow region*, which is defined as the set of states which can be driven to the target region with the same control policy, and gives sufficient conditions for a set of invariant manifolds to bound common flow regions. In this paper, such invariant manifolds are determined by appropriate Lyapunov functions. The switchings are allowed to occur only if the continuous state lies on the intersection of those invariant manifolds. Since the switching logic is described by a Petri net, only sequences of invariant manifolds that satisfy the discrete specifications have to be considered.

The paper is organized as follows. Section 2 presents programmable timed Petri nets which are used in section 3 to model hybrid control systems. In Section 4 we discuss in detail a Petri net approach to hybrid control which emphasizes supervisory control of hybrid systems and we give a simple illustrative example. Note that related work has appeared in [23,24].

2 Programmable Timed Petri Nets

Programmable timed Petri nets were introduced in [17] and are used to generate the switching logic of the hybrid system. In particular, a programmable timed Petri net (PTPN) is a timed Petri net whose places, transitions, and arcs are all labeled with formulae representing constraints and reset conditions on the rates and times generated by a set of continuous-time systems called *clocks*. The model can be seen as an extension of the Alur-Dill hybrid automaton model [25,26].

An *ordinary Petri net structure* [27,28,29] is the 4-tuple $\mathcal{N} = (P, T, I, O)$ where P is a finite set of places, T is a finite set of transitions, $I \subset P \times T$ is a set of input arcs (from places to transitions), and $O \subset T \times P$ is a set of output arcs (from transitions to places). The *preset* and *postset* of a place p are defined by $\bullet p = \{t \mid (t, p) \in O\}$ and $p\bullet = \{t \mid (p, t) \in I\}$. The *preset* and *postset* of a transition t are defined similarly as $\bullet t = \{p \mid (p, t) \in I\}$ and $t\bullet = \{p \mid (t, p) \in O\}$.

The *marking* of a Petri net is a mapping $\mu : P \rightarrow \mathbb{Z}^+$ from the set of places onto the nonnegative integers which assigns to each place p a number of tokens $\mu(p)$. The marking can be represented also by an n_p -dimensional vector $\mu = (\mu_1, \dots, \mu_p)$, where $n_p = |P|$. The vector μ gives for each place p_i , the number of tokens in that place, $\mu_i = \mu(p_i)$. To avoid confusion, the marking μ is interpreted as a mapping when it is appeared with an argument and as a vector of nonnegative integers otherwise. The dynamics of ordinary Petri nets are characterized by the evolution of the marking vector which is referred to as the *state* of the net.

The transition t is *enabled* when each one of its input places is marked with at least one token, $\mu(p) > 0$ for all $p \in \bullet t$. An enabled transition may fire. The transition t *fires* by removing one token from each one of its input places and by placing one token to each one of its output places. If $\mu(p)$ and $\mu'(p)$ denote

the marking of place p before and after the firing of enabled transition t , then

$$\mu'(p) = \begin{cases} \mu(p) + 1 & \text{if } p \in t \bullet \setminus \bullet t \\ \mu(p) - 1 & \text{if } p \in \bullet t \setminus t \bullet \\ \mu(p) & \text{otherwise} \end{cases} \quad (1)$$

The firing of the transition t is described by the firing function $q : T \rightarrow \{0, 1\}$ such that $q(t) = 1$ if t is firing and $q(t) = 0$ otherwise. In untimed Petri nets one can prohibit controlled transitions from firing, but cannot force the firing of a transition at a particular instant. In a timed Petri net controlled transitions are forced to fire, as this can be accomplished by considering the firing functions to be functions of a global time. For the timed Petri net, the firing of a transition occurs over a time interval $[\tau_0, \tau_f]$. The length of this interval is called the transition's *holding time*. A transition t which starts to fire at time τ_0 is said to be *committed*. During the time that the transition is committed, the network's marking vector is not changed. It is only when the firing is completed at time τ_f that the marking vector is changed according to equation (1) given above.

The holding times can be seen as control variables. They can be controlled by specifying conditions which cause transitions to fire. The conditions that characterize the holding times are represented by logical propositions defined over a set of vector dynamical equations, which can be seen as a set of *local clocks*.

Consider the set, $\mathcal{X}$, of N *local clocks* where the i th clock $\mathcal{X}_i$ is denoted by the triple $(\dot{x}_i, x_{i0}, \tau_{i0})$. $x_{i0} \in \mathbb{R}^n$ is a real vector representing the clock's offset. τ_{i0} is an initial time (measured with respect to the global clock) indicating when the local clock was started. $\dot{x}_i : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is a Lipschitz continuous automorphism over $\mathbb{R}^n$ characterizing the local clock's rate. It is assumed that the clock rate $\dot{x}_i$ is denoted by the automorphism f . The *local time* generated by the i th clock will be denoted as x_i which is a continuous function over $\mathbb{R}^n$ that is the solution to the following initial value problem for $\tau > \tau_{i0}$,

$$\frac{dx_i}{dt} = f(x_i) \quad (2)$$

$$x_i(\tau_{i0}) = x_{i0}. \quad (3)$$

The *state* of the i th timer is the ordered pair $z_i(\tau) = (x_i(\tau), \dot{x}_i(t))$. The ensemble of all the local clock states will denoted by $z(\tau)$.

The interval $[\tau_0, \tau_f]$ over which a transition t will be firing is characterized by conditions on the "local time" $x_i(\tau)$ and the "clock rate" $\dot{x}_i(t)$ of the i th timer. These conditions are described by formulae in a propositional logic whose atomic formulas are equations over the local times or clock rates of $\mathcal{X}$. In the next section, the local clocks are used to describe the continuous dynamics of the hybrid systems at each operational mode. The hybrid system switches modes based on constraints on the continuous states. The atomic and the propositional formulas which will be used to describe the conditions on the states of the vector dynamical equations are defined next. Their form is general enough to describe a variety of constraints which will be used to characterize the evolution of the hybrid system in the next section.

Definition 1. An atomic formula, p , takes one of the following forms;

1. It can be a time constraint of the form $h(x_i) = 0$ or $h(x_i) < 0$ where $h : \mathbb{R}^n \rightarrow \mathbb{R}$ is a real valued function. This formula means that the clock time x_i satisfies the equation.
2. The atomic formula p can be a rate constraint of the form $\dot{x}_i = f$ which means that the i th clock's rate $\dot{x}_i$ is equal to the vector field $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$.
3. Finally, p can be a reset equation of the form $x_i(\tau) = \bar{x}_0$ which says that the i th clock's local time at global time τ is set to the vector $\bar{x}_0$.

Definition 2. A well-formed formula or WFF is defined as any expression generated by a finite number of application of the following rules;

1. Any atomic formula is a WFF,
2. If p and q are WFFs, then $p \wedge q$ is a WFF.
3. If p is a WFF, then $\neg p$ is a WFF.

The set of all WFFs formed in this manner will be denoted as $\mathcal{P}$. Consider an ordinary Petri net, $\mathcal{N} = (P, T, I, O)$ and a set of logical timers, $\mathcal{X}$. A *programmable timed Petri net* (PTPN) is denoted by the ordered tuple, $(\mathcal{N}, \mathcal{X}, \ell_P, \ell_T, \ell_I, \ell_O)$ where the functions $\ell_P : P \rightarrow \mathcal{P}$, $\ell_T : T \rightarrow \mathcal{P}$, $\ell_I : I \rightarrow \mathcal{P}$, and $\ell_O : O \rightarrow \mathcal{P}$ label the places, transition, input arcs, and output arcs (respectively) of the Petri net $\mathcal{N}$ with WFFs in $\mathcal{P}$.

The *syntax* for well formed formulas is defined with respect to the underlying Petri net structure of the form $\mathcal{N} = (P, T, I, O)$ and the set of local clocks $\mathcal{X}$. The local clock state z at time τ is said to *satisfy* a formula $p \in \mathcal{P}$ if p is “true” for the given clock state, $z(\tau)$. The truth of the well-formed formulas is understood in the usual sense.

3 Hybrid Control Systems

In this section, programmable timed Petri nets are used to model hybrid systems. The hybrid control systems of interest in this paper are described by the following equations

$$\dot{x} = f_{i(t)}(x(t)) \quad (4)$$

$$i(t) = q(x(t), i(t^-)) \quad (5)$$

where $x(t) : \mathbb{R} \rightarrow \mathbb{R}^n$ and $i : \mathbb{R} \rightarrow \mathbb{Z}^+$ denote the continuous and discrete states of the system, respectively. The continuous dynamics are controlled by a finite collection of N control strategies

$$\mathcal{D} = \{f_1, f_2, \dots, f_N\} \quad (6)$$

where $f_i : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$ for $i \in \{1, \dots, N\}$ are locally Lipschitz continuous functions. The discrete state of the system is controlled by a *successor* function

$q : \mathbb{R}^n \times \mathbb{Z}^+ \rightarrow \mathbb{Z}^+$ which determines the next possible discrete state $i(t)$ at time t given the current continuous state and the “previous” discrete state $i(t^-)$.

We introduce now some additional notation that will be useful in formulating the control algorithms later in the section. The firing times of transition t are described by $\sigma^t(n)$, $n \in \mathbb{Z}^+$, where $\sigma^t(k) \in \mathbb{R}$ represents the duration of the k^{th} firing of transition t . During the time interval $\sigma^t(k)$ the tokens of the input places of transition t do not change. These tokens are put into the output places of t upon the completion of firing of the transition, according to the enabling condition of the untimed Petri net. We assume that at each time instant exactly one transition is firing. In addition, we assume that $0 < \Delta \leq \sigma^t(n) < \infty$, for some $\Delta \in \mathbb{R}$, for all firings n and transitions t . We may easily incorporate in our model instantaneous transitions, but these correspond to jumps in the continuous state and will not be considered here. The assumption $0 < \Delta \leq \sigma^t(n)$ eliminates the possibility of infinitely many switchings in a finite time interval.

The control algorithm for the mode selection problem will be based on structural information associated with the places of the Petri net. Routing policies for timed Petri nets are used usually for resolution of conflicts and were introduced in [30]. In our case, we define a mapping $\nu^p(n) : \mathbb{Z}^+ \rightarrow T$ for each place $p \in P$, where $\nu^p(k)$ identifies the particular transition $t \in p\bullet$ to which the k^{th} token to enter place p is to be routed. Note that more than one transition is enabled but only one is allowed to actually fire. If the k^{th} token is routed to $t \in p\bullet$, then the transition t wins the token, which after a firing time of $\sigma^t(k)$ is routed to $t\bullet$, the output places of the transition.

Next, a firing event is defined as the pair (t, τ) which denotes that the transition t starts firing at time τ . Consider the sequence of firing events

$$s = (t_{i_0}, \tau_0), (t_{i_1}, \tau_1), \dots, \quad i_j \in \{1, \dots, N\}, \text{ for } j = 0, 1, 2, \dots \quad (7)$$

where j denotes the ordering of the transitions that fire. For example $s = (t_1, \tau_0), (t_3, \tau_1), \dots$ denotes that t_1 fires at τ_0 , next t_3 fires at τ_1 and so on. The firing time intervals are defined by the equation

$$\sigma^{t_{i_j}}(k) = \tau_{k+1} - \tau_k \quad (8)$$

At the k^{th} firing of the network, the transition t_{i_j} starts firing (at time τ_k) for $\sigma^{t_{i_j}}(k)$ time units (until τ_{k+1}). The continuous state of the system during this interval evolves according to

$$\dot{x}(\tau) = f_{i_j}(x(\tau)), \text{ for } \tau_k \leq \tau < \tau_{k+1}. \quad (9)$$

The *event projection* and the *timed projection* of the sequence s are defined as

$$\pi_1(s) = i_0, i_1, i_2, \dots \quad (10)$$

$$\pi_2(s) = \sigma^{t_{i_0}}(k_0), \sigma^{t_{i_1}}(k_1), \dots \quad (11)$$

These are used later in this section.

4 Supervisor

The supervisor has two main tasks. The first task is to allow only sequences of events that satisfy specifications imposed on the discrete-event part of the hybrid plant. In particular, consider the net $\mathcal{N}$ of the hybrid system. The objective here is to restrict the possible mode switches of the systems to satisfy additional logical constraints (for example mutual exclusion constraints) that have not been taken into consideration in the modeling phase of the hybrid plant. This can be accomplished without any information about the continuous dynamics. The differential equations of the continuous subsystems associated with the transitions are used to label these transitions. The second task of the supervisor is to enforce firing times that satisfy specifications on the continuous state of the plant. In untimed Petri nets one can prohibit transitions from firing, but cannot force the firing of a transition at a particular instant. In a timed Petri net controlled transitions are forced to fire, as this can be accomplished by considering the firing vectors to be functions of the global time τ . We will show that for a special class of problems, we can first determine the routing policy and then the firing times that will not violate certain conditions imposed on the continuous dynamics. These conditions will be expressed as well formed formulas labeling the input and output arcs of the Petri net.

4.1 Supervisor Control of Petri Nets Based on Place Invariants

The first step is to satisfy the discrete specifications of the hybrid plant by applying DES control methods. We assume that the discrete specifications are described by linear inequalities on the marking vector of the Petri net. A methodology for DES control based on Petri net place invariants has been proposed in [18]. A feedback controller based on place invariants is implemented by adding control places and arcs to existing transitions in the Petri net structure. Although the method was developed for ordinary Petri nets, the introduction of time delays associated to each transition will not affect the controlled behavior of the Petri net with respect to the discrete specifications. The supervisor is used to enforce a set of linear constraints on the discrete state of the hybrid plant. These constraints can describe a broad variety of problems including forbidden state problems, mutual exclusion problems, a class of logical predicates on plant behavior [31], conditions involving the concurrence of events, and the modeling of shared resources.

The system to be controlled is the untimed Petri net $\mathcal{N} = (P, T, I, O)$, which is called the plant net. We assume that the plant net has n places and m transitions and its incidence matrix is D_p . The controller net is a Petri net with incidence matrix D_c made up of the transitions of the plant net and a separate set of places. The controlled net is the Petri net with incidence matrix D made up of both the plant and the controller net. The control objective is to enforce the discrete state to satisfy constraints of the form

$$L\mu_p \leq b \quad (12)$$

where μ_p is the marking vector of the plant net, L is an $n_c \times n$ integer matrix, b is an $n_c \times 1$ integer vector, and n_c is the number of 1-dimensional constraints of the type $\sum_{i=1}^{n_p} l_i \mu_i \leq \beta$.

This inequality constraint can be transformed to an equality by introducing an external Petri net controller that contains places representing nonnegative slack variables. Then

$$L\mu_p + \mu_c = b \quad (13)$$

where μ_c is an n_c integer vector which represents the marking of the controller places. The structure of the controller net will be computed by observing that the introduction of the slack variables forces a set of place invariants on the controlled system. A *place invariant* is defined by an integer vector x that satisfies

$$x^T \mu = x^T \mu_0 \quad (14)$$

where μ_0 is the initial marking and μ any reachable subsequent marking. The place invariants of a net are elements of the kernel of the net's incidence matrix, and they can be computed by finding integer solutions to

$$x^T D = 0 \quad (15)$$

where D is an $n \times m$ incidence matrix. The matrix D_c contains the arcs that connect the controller places to the transitions of the plant net. The incidence matrix D of the closed loop system is given by

$$D = \begin{bmatrix} D_p \\ D_c \end{bmatrix} \quad (16)$$

and the marking vector μ and the initial marking μ_0 are given by

$$\mu = \begin{bmatrix} \mu_p \\ \mu_c \end{bmatrix} \quad \mu_0 = \begin{bmatrix} \mu_{p_0} \\ \mu_{c_0} \end{bmatrix} \quad (17)$$

Note that equation (13) is in the form of (14), thus the invariants defined by equation (13) on the system (16),(17) must satisfy equation (15).

$$X^T D = [L, I] \begin{bmatrix} D_p \\ D_c \end{bmatrix} = 0 \quad (18)$$

$$LD_p + D_c = 0 \quad (19)$$

If D_c is chosen as the solution of equation (19), then the rows of $[L, I]$ are elements of the kernel of the net's incidence matrix. Therefore, they represent place invariants of the closed loop systems and equation (13) is satisfied. Since $\mu(p) > 0$ for all $p \in P$, inequality (12) holds componentwise. The above analysis leads to the following proposition presented in [32].

Proposition 1. *The Petri net controller with incidence matrix D_c and initial marking μ_{c_0} , which enforces the constraints $L\mu_p \leq b$ when included in the closed loop system (16) with marking (17) is defined by*

$$D_c = -LD_p \quad (20)$$

with initial marking

$$\mu_{c_0} = b - L\mu_{p_0} \quad (21)$$

assuming that the transitions with arcs from D_c are controllable, observable, and that $\mu_{c_0} \geq 0$.

This proposition designs a controller that enforces the linear constraints $L\mu \leq b$ under the assumption that the controller will enable or inhibit only controllable and observable transitions. These results have been extended for handling uncontrollable and unobservable transitions in [33]. In the hybrid systems case, we have associated transitions to continuous subsystems described by differential equations. It is assumed that the supervisor can force and observe the firing of the transitions. This is accomplished by imposing conditions described by well-formed formulas on the input and output arcs of the transitions, as described in the next section.

4.2 Hybrid Strategy based on Equilibria

In the nonlinear control literature, switching has been used to expand the domain of attraction of control systems [19,20]. In the hybrid systems case, we assume that the continuous part admits a family of equilibria corresponding to different symbolic inputs generated by the discrete event part. Each equilibrium has a domain of attraction associated with it. The idea is to switch at discrete time instants from one symbolic input to another in a way that the system gradually progresses from one equilibrium to another towards the final equilibrium.

This idea can be formalized using an *invariant based approach* for hybrid systems proposed in [21,22]. A *common flow region* for a given target region, is defined as a set of states which can be driven to the target region with the same control policy. The approach as described by Stiver *et al.* considers common flow regions which are bounded by *invariant hypersurfaces*, *cap boundaries* and an *exit boundary*. Invariant hypersurfaces and cap boundaries which are described next in the section, form manifolds to bound a region so that the state trajectory can leave the region only through the exit boundary. In [21] sufficient conditions for a set of hypersurfaces to form a common flow region were established. Here, a Lyapunov approach is followed to efficiently compute hypersurfaces that form common flow regions for each control policy. Each common flow region is identified as a subset of an invariant manifold defined by a Lyapunov functional and is associated with a control policy. Since the switching function is generated by the underlying Petri net only sequences of invariant manifolds that correspond to control policies which satisfy the discrete specifications have to be examined.

Definition 3. For the continuous part of the hybrid plant, the set B is a common flow region for a given region R if

$$\forall x(t_0) \in B, \exists t_1, t_2, t_0 < t_1 < t_2$$

such that

$$x(t) \in B, t \leq t_1$$

and

$$x(t) \in R, t_1 < t < t_2$$

subject to

$$\dot{x}(t) = f_i(x(t))$$

In [21] two proposition are given which provide sufficient conditions for a set of hypersurfaces to form a common flow region. These hypersurfaces can be either *invariant* under the vector field of the given control policy or *cap boundaries* for the given vector field. Invariant hypersurfaces and cap boundaries form manifolds to bound a common flow region, so that the state trajectories cannot cross those manifolds.

Definition 4. A set $M \subset X$ is said to be invariant with respect to the system $\dot{x} = f(x)$ if $x(t_0) \in M \Rightarrow x(t) \in M, \forall t \in \mathbb{R}$.

Consider the set of smooth hypersurfaces, $\{h_i, i \in I_B \subset 2^I\}$. The hypersurfaces is a set of smooth functionals $\{h_i : \mathbb{R}^n \rightarrow \mathbb{R}, i \in I_B\}$, defined on the state space of the plant. Each functional must satisfy the condition

$$\nabla_x h_i(\xi) \neq 0, \forall \xi \in \mathcal{N}(h_i) \quad (22)$$

which ensures that the null space of the functional $\mathcal{N}(h_i) = \{\xi \in \mathbb{R}^n : h_i(\xi) = 0\}$ forms an $n - 1$ dimensional manifold separating the state space.

Invariant hypersurfaces For a hypersurface h_i to be invariant under the vector field f of the given control policy, the following condition must be satisfied

$$\nabla_\xi h_i(\xi) \cdot f(\xi) = 0 \quad (23)$$

The set of all invariant hypersurfaces can be found in terms of $n - 1$ functionally independent mappings which form the basis for the desired set of functionals, $\{h_i\}$. This basis is obtained by solving the characteristic equation

$$\frac{dx_1}{f_1(x)} = \frac{dx_2}{f_2(x)} = \dots = \frac{dx_n}{f_n(x)} \quad (24)$$

where $f_j(x)$ is the j th element of $f(x)$ ($f(x)$ is used rather than $f_i(x)$ to avoid subscript confusion).

Cap boundaries For a hypersurface h_c to form a cap boundary for a given vector field f and common flow region B , the following condition must be satisfied

$$\nabla_\xi h_c(\xi) \cdot f(\xi) < 0, \forall \xi \in B \cap \mathcal{N}(h_c) \quad (25)$$

Consider the hypersurface $h_c(x)$ that forms a cap boundary for the common flow region B . Assume that there exists an appropriate Lyapunov function $V(x)$ for the vector fields f such that

$$\begin{aligned} V(x) &> 0, \forall x \in B \\ V(x) &\rightarrow \infty \text{ as } \|x\| \rightarrow \infty \\ \dot{V}(x) &< 0, \forall x \in B \end{aligned} \quad (26)$$

then $\Omega_c = \{x \in \mathbb{R}^n \mid V(x) \leq c\}$ is bounded and the hypersurface $h_c(x) = V(x) - c$ is a cap boundary candidate. The constant parameter c can be selected appropriately so that the hypersurface $h_c(x)$ which bounds the common flow region B satisfies certain safety constraints.

As it was recognized in [21], the task of determining suitable invariant hypersurfaces is very difficult in general; for special cases (e.g. integrator systems), the differential equation (24) was solved analytically, otherwise an algorithm which was computationally inefficient was used. Here, we introduce a Lyapunov approach to determine cap boundaries. This approach is more efficient and can be applied to a larger class of systems; furthermore, the design based on Lyapunov functions exhibits desirable robustness properties. However, by assuming that the common flow regions are bounded by manifolds defined by Lyapunov functionals, we impose restrictive conditions on the dynamics of the continuous subsystems. In most of the cases, these conditions are quite restrictive but they suggest a systematic way to compute common flow regions. For example, we require stability (in the region of interest) for the continuous subsystems because then we can systematically approximate the region of attraction of an equilibrium. The next proposition gives sufficient conditions for the state to progress from one equilibrium to another.

Proposition 2. *Let $f_{i_1}, f_{i_2} \in \Sigma$ satisfy the following assumptions*

1. *Each f_i is globally Lipschitz and admits an isolated equilibrium point $\bar{x}_i$, and $\bar{x}_i$ is asymptotically stable w.r.t. f_i .*
2. *For each f_i there exists an appropriate Lyapunov function $V_i : \mathbb{R}^n \rightarrow \mathbb{R}$ and $\Omega_i = \{x \in \mathbb{R}^n \mid V_i(x) \leq c_i\}$ such that*

$$\begin{aligned} V(x) &> 0, \forall x \in \Omega_i \\ V(x) &\rightarrow \infty \text{ as } \|x\| \rightarrow \infty \\ \dot{V}(x) &< 0, \forall x \in \Omega_i \end{aligned} \tag{27}$$

In addition, assume that $\Omega_{i_1} \cap \Omega_{i_2} \neq \emptyset$ and $\bar{x}_{i_1} \in R' = \text{int}(\Omega_{i_1} \cap \Omega_{i_2})$, then for every $x_0 \in \Omega_{i_1}$ there exists a switching sequence

$$s(x_0, t_0) = (i_1, \sigma^{t_{i_1}}(k_0)), (i_2, \sigma^{t_{i_2}}(k_1))$$

which drives the state to a region R of the equilibrium point $\bar{x}_{i_2}$.

Proof. Let $\Omega = \Omega_{i_1} \setminus \Omega_{i_2}$ and define the hypersurface $h_c(x) = \partial\Omega \cap \partial\Omega_{i_1}$ and $h_e(x) = \partial\Omega \cap \partial\Omega_{i_2}$. Since $\bar{x}_{i_1} \in R'$ is an asymptotically equilibrium point for f_{i_1} , Ω is a common flow region for $R' = \text{int}(\Omega_{i_1} \cap \Omega_{i_2})$. Let $\Omega' = \Omega_{i_2} \setminus R$ and define the hypersurface $h_c(x) = \partial\Omega_{i_2}$ and $h_e(x) = \partial R$, then Ω' is a common flow region for the target region R .

If we assume that the switching logic of the hybrid system is described by a DES (here a programmable timed Petri net), only sequences of vector fields that correspond to control policies that satisfy the discrete specifications have

to be considered. The control policies that satisfy the discrete specifications are exactly those that are accepted by the controlled Petri net which includes also the supervisor designed using the methodology based on place invariants (see above). They can be determined by identifying the periodic behavior of the Petri net. Several methods have been proposed to determine the periodic behavior of Petri nets (reachability tree, transition invariants, unfolding). For this problem, the size of the Petri net depends on the number of available symbolic inputs, hence we can efficiently identify all the periodic behaviors by computing the fundamental cycles of the reachability tree.

The underlying Petri net structure, which generates the switching policy offers two important advantages. First, it makes possible to efficiently design the supervisor that satisfy specifications that frequently appear in complex systems such as generalized mutual exclusion constraints. Second, it reduces considerably the search for common flow regions, since only desirable switching strategies generated by the controlled Petri net have to be examined.

The following corollary gives sufficient conditions for a switching sequence generated by the controlled Petri net to drive the continuous state x_0 to a target region of the state space. It is assumed that the initial conditions belong to the region of attraction Ω_{i_0} of the first control policy and that it progresses towards $\bar{x}_{i_m} \in \Omega_{i_m}$ by allowing switchings to occur on the intersection $\Omega_{i_j} \cap \Omega_{i_{j+1}}$ of consecutive invariant manifolds. In the case when all the pairs of control policies satisfy Proposition 2, the set $\Omega_{i_j} \cap \Omega_{i_{j+1}}$ will be nonempty and the proof is clear.

Corollary 1. *Suppose there exists a switching sequence with event projection $\pi_1(s) = i_0, i_1, \dots, i_m$ accepted by the controlled Petri net such that every pair $(f_{i_j}, f_{i_{j+1}})$ satisfies Proposition 1. Given a target region R such that $\bar{x}_{i_m} \in \text{int}(R)$, there exists routing policy $\nu^p(n)$ to drive the continuous state from any initial condition $x_0 \in \Omega_{i_0}$ to the region R in finite time. The firing time intervals $\sigma^t(n)$ will be chosen so that the switchings occur while $x \in \text{int}(\Omega_{i_j} \cap \Omega_{i_{j+1}})$.*

Remark The condition that every pair $(f_{i_j}, f_{i_{j+1}})$ satisfies Proposition 2 can be relaxed by allowing intermediate transitions which will keep the continuous state in the domain of attraction of $\bar{x}_{i_{j+1}}$.

The supervisor is implemented by assigning well-formed formulas to the places, transitions, input and output arcs of the controlled Petri net. Let $\{h_i\}$, $i = 1, \dots, n$ be the set of hypersurfaces that bound a region M of the state space X . We can use the following well-formed formulas to describe that $x \in M$.

$$\ell = p_1 \wedge p_2 \wedge \dots \wedge p_n \quad (28)$$

where p_i is a constraint of the form $h_i(x) < 0$. Consider a pair of vector fields $(f_{i_j}, f_{i_{j+1}})$ that satisfy Proposition 2 and let B_{i_j}, R_{i_j} and $B_{i_{j+1}}, R_{i_{j+1}}$ be the corresponding common flow and target regions. From Proposition 2 we have that the target region R_{i_j} coincides with the common flow region $B_{i_{j+1}}$. The switching algorithm is implemented by the following labeling functions, where $p \in P$ is the place to connect the output arc of t_{i_j} to the input arc of $t_{i_{j+1}}$

1. $\ell_P(p)$ is chosen to be a tautology.
2. $\ell_T(t_{i_j}), \ell_T(t_{i_{j+1}})$ are chosen to be the atomic rate formulas $\dot{x} = f_{i_j}(x)$ and $\dot{x} = f_{i_{j+1}}(x)$ respectively.
3. $\ell_O((t_{i_j}, p))$ is chosen to be a WFF of the form (28) representing that $x \in R_{i_j}$.
4. $\ell_I((p, t_{i_{j+1}}))$ is chosen to be a WFF of the form (28) representing that $x \in B_{i_{j+1}}$.

Assuming that transition t_{i_j} is firing, the next transition to fire, $t_{i_{j+1}}$ is determined by the routing policy $\nu^p(n)$ so that the pair $(f_{i_j}, f_{i_{j+1}})$ satisfies Proposition 2. Transition $t_{i_{j+1}}$ will fire only when the firing time intervals $\sigma^t(n)$ assign true values to the logic formulas $\ell_O((t_{i_j}, p))$ and $\ell_I((p, t_{i_{j+1}}))$. For the initialization of the hybrid system we assume that $\ell_I((p, t_{i_0}))$ is a tautology.

Remark In the case when the Petri net is live and the event projection generated by the controlled Petri net $\pi_1(s)$ is an infinite sequence that satisfies Corollary 1, the hybrid system exhibits a periodic behavior in the sense that the continuous state is visiting periodically neighborhoods of the equilibria.

Affine Systems A class of systems that satisfy the conditions for supervisory control design of the previous section is the affine systems. They represent physical systems that are described by linear ordinary differential equations with one additional assumption. The input is allowed to take a finite number of pre-specified constant values.

Consider the case the continuous dynamics are described by

$$\dot{x} = Ax + c_i, \quad T_k \leq t < T_{k+1}$$

where $c_i \in W \subset \mathbb{R}^n$ a finite set of control vectors and the matrix $A \in \mathbb{R}^{n \times n}$ is Hurwitz.

Let $f_i(x) = Ax + c_i$, then $\bar{x}_i = -A^{-1}c_i$ is a globally asymptotically stable equilibrium point for $\dot{x} = f_i(x)$. In view of the global asymptotic stability of each equilibrium point, it is clear that Proposition 2 holds for every pair of control inputs. The values for the control input can be selected so that the continuous state can be driven to prescribed regions of the state space.

Example: Hybrid System Describing Resource Contention Consider the case of two different processes that use the same resource to carry out their operations. This is a conflict situation which stems from the resource contention. More specifically, assume that each process consists of two different operations which are described by ordinary differential equations and the switching policy is represented by the Petri net in Fig. 1. This situation arises frequently in physical systems when different processes share the same resources. We will use this Petri net to describe the switching policy for two examples that follow. The first example is a temperature control system where the continuous dynamics are described by affine systems. In the second example, we consider a hybrid system with continuous dynamics described by first order integrators.

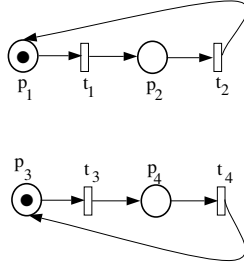


Fig. 1. Petri net describing the switching policy of the hybrid plant.

The incidence matrix of the plant net is

$$D_p = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \quad (29)$$

and the initial condition of the marking vector $\mu_{p_0} = [1, 0, 1, 0]^T$. We consider the mutual exclusion constraint $L\mu_p \leq b$, where $L = [0, 1, 0, 1]$ and $b = 1$. Using Proposition 1 the closed loop system has the incidence matrix

$$D = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 \\ -1 & 1 & -1 & 1 \end{bmatrix} \quad (30)$$

and initial condition $\mu_0 = [1, 0, 1, 0, 1]$. The last row of the incidence matrix D represents the Petri net supervisor.

The controlled Petri net is shown in Fig. 2 and describes the switching policy for the hybrid system which satisfies the mutual exclusion constraint.

Temperature Control System Let a temperature control system be described by the electrical circuit shown in figure 3. Here, an electrical analog of the temperature control system is used by considering the temperature being analogous to electric voltage, heat quantity to current, heat capacity to capacitance, and thermal resistance to electrical resistance. The control objective is to control the temperature at a point at the system by applying the heat input at a different point. The temperature control example is used in [34] to illustrate PID control design. Here, we assume that only discrete levels are available for the current (heat) input (u).

Let x_1 and x_2 denote the voltages across the capacitors C_1 and C_2 respectively. Suppose that the (voltages) temperatures x_1 and x_2 are to be controlled

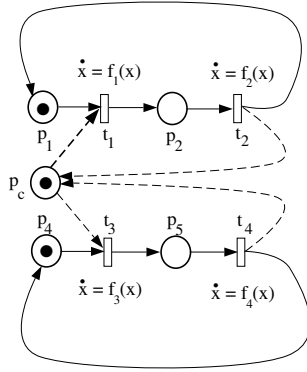


Fig. 2. The controlled Petri net of the resource contention example

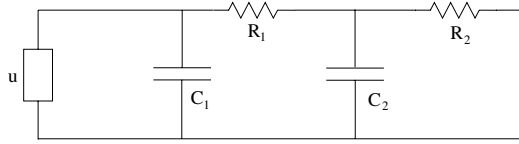


Fig. 3. The temperature control system

by changing the (current) heat input u , which is allowed to take finitely many discrete values. Consider that the numerical values of the electrical elements are

$$\begin{aligned} R_1 &= R_2 = 1 \\ C_1 &= C_2 = 1 \end{aligned}$$

Then (using Kirchhoff's laws) the system is described by the state-space equation

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -1 & 1 \\ 1 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u \quad (31)$$

Assume that q_i , $i = 1, \dots, N$ are the available discrete levels for the input that correspond to the available control policies. It is easily verified that the matrix A is Hurwitz and therefore $\bar{x}_i = -A^{-1}c_i$ where

$$A = \begin{bmatrix} -1 & 1 \\ 1 & -2 \end{bmatrix}, \quad c_i = \begin{bmatrix} q_i \\ 0 \end{bmatrix}$$

is a globally asymptotically stable equilibrium point for the system $\dot{x} = Ax + c_i$.

We assume that the discrete levels of the heat input are $q_1 = -10$, $q_2 = 10$, $q_3 = -20$, and $q_4 = 20$ and that the switching policy is described by the Petri net in Fig. 2 representing resource contention, where transition t_i corresponds to the control input c_i . In this case, the supervisor will determine the routing policy and the firing time intervals so that the continuous state of the hybrid

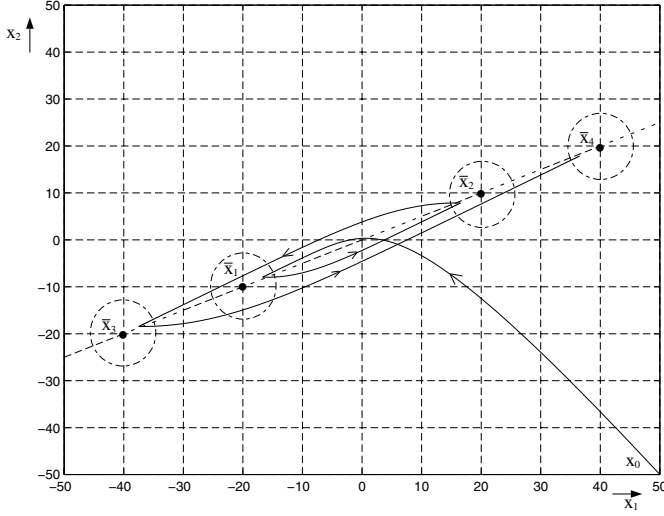


Fig. 4. Periodic behavior of the hybrid system

system visits periodically neighborhoods of the equilibria (Fig. 4). We consider a ball $B_i(\bar{x}_i, r)$ of radius r centered at $\bar{x}_i$, $i = 1, 2, 3, 4$ and we label the input and output arcs of the transitions with the WFFs $\ell_O((t_i, p)) = \ell_I((p, t_{i+1})) = x \in B_i(\bar{x}_i, r)$. Then the supervisor allows switchings to occur only when the above logical propositions are true.

5 Supervisory Control Design for First Order Integrators

Recently, attention has been focused on a particular class of hybrid systems in which the continuous dynamics are governed by the differential equation $\dot{x}(t) = c$, where $c \in \mathbb{R}^n$ [35,36,37,38]. In [36] hybrid systems with continuous dynamics described by first order integrators are used for the control of batch processes. In the case when the continuous dynamics are described by first order integrators, the previous algorithm cannot be applied since the continuous part does not admit any isolated equilibria. In the following, we present an algorithm based on a similar idea as in the case of multiple equilibria. For each symbolic input there exist a family of invariant sets. The switching from a symbolic input to another occur at discrete time instants in a way that the system gradually progresses towards the target region. We determine a sequence of events that drive the state to the prescribed target region by solving a linear programming problem as in [38].

5.1 Hybrid Plant

The continuous state of the hybrid plant evolves in $X \in \mathfrak{R}^n$ and is described by first order integrators

$$\dot{x}(t) = c_i, \quad T_k \leq \tau_i \leq T_{k+1}$$

where $c_i \in W \subset \mathfrak{R}^n$ a finite set of control vectors and T denotes the global time.

It is assumed as in the section 3 that the switching strategy is embedded in a programmable timed Petri net. Each transition is associated with a control policy. This assignment is defined by the labeling function $\ell_T(t) : T \rightarrow \Sigma$ which is chosen to be an atomic rate formula of the form $\dot{x} = c_i$.

Definition 5. A set $\mathcal{C}$ is said to be a finitely generated cone if it has the form

$$\mathcal{C} = \left\{ x : x = \sum_{j=1}^r \tau_j c_j, \tau_j \geq 0, c_j \in \mathfrak{R}^n, j = 1, \dots, r \right\}$$

In the following, we assume that the finitely generated cone by the set W coincides with the continuous state space X . This assumption guarantees that continuous specifications such as state targeting can be satisfied everywhere in the state space X .

The system $\dot{x} = c$ admits a family of invariant sets described by the equation $x = ct + x_0$, which represents a family of parallel lines parameterized in t . The path from the initial state to the target region will be found by solving a linear programming problem to determine the time interval the state of the system evolves in each particular invariant set.

Supervisor The procedure for the design of the supervisor is similar to that of the previous section. First, applying DES control methods we construct a controlled Petri net which satisfies the discrete specifications. Assume now that the control objective is to drive the state to the target region $\mathcal{T} \subset X$. The task of the supervisor is to select at time T_k a control policy accepted by the Petri net, and to decide for how long it should be active. A systematic way to determine the temporal and routing data is first to identify all the periodic behaviors the controlled Petri net can generate and then assign temporal constraints to the switching times.

Proposition 3. Consider the switching sequence

$$s(t_0, x_0) = \{(i, \sigma^{t_i}(k))\}_{i=1, \dots, N, k=0, 1, \dots, m}, \quad k_0 = t_0$$

such that

1. The event projection $\pi_1(s) = i_0, i_1, i_2, \dots, i_m$ consists of the transitions which form all the fundamental cycles.

2. The timed projection $\pi_2(s) = \sigma^{t_{i_0}}(k_0), \sigma^{t_{i_1}}(k_1), \dots, \sigma^{t_{i_m}}(k_m)$ satisfies the following conditions $\sum_k \sigma^{t_i}(k) = \tau_i$ and $0 < \Delta \leq \sigma^t(n) < \infty$ for all transitions and firings; $\tau = [\tau_1, \dots, \tau_N]$ is the solution of the following linear programming problem

$$\begin{aligned} & \min a^T \tau \\ & \text{subject to : } \begin{cases} x_f = x_0 + \sum_{i=1}^N \tau_i c_i \in \mathcal{T} \\ \tau_i \geq \Delta, i = 1, \dots, N \end{cases} \end{aligned}$$

where τ is the vector of operation times to be determined, a is a weighting vector, x_f is the response of the continuous part at the time instant when the switching sequence $s(t_0, x_0)$ is completed.

Then, the continuous state $x \in \mathbb{R}$ is driven to the target region in finite time $t = \sum_{i=1}^N \tau_i$.

Proof. By integrating the state equation (5.1) all the reachable states x_f from initial state x_0 are given by

$$x_f = x_0 + \sum_{i=1}^N \tau_i c_i \quad (32)$$

where τ_i represents the total time the corresponding control policy c_i is active. Although there is no unique switching policy to satisfy the convex constrain $x_f \in \mathcal{T}$, the solution of the linear programming problem is unique and gives the control law that drives the state to the target region in minimum time.

Additionally, the necessary number of switches can be minimized by considering one firing of each transition. Additional safety conditions expressed as convex constraints can be incorporated in the linear programming problem. The supervisor solves the above linear programming problem and labels the places, transitions, input and output arcs of the controlled Petri net as follows

1. $\ell_P(p)$ is chosen to be a tautology.
2. $\ell_T(t_i)$ is chosen to be the atomic rate formulas $\dot{x} = c_i$.
3. $\ell_I((p, t_i))$ and $\ell_O((t_i, p))$ are chosen to be time constraints implement by a local timer so that $\sum_k \sigma^{t_i}(k) = \tau_i$.

Example: First Order Integrators We assume now that the continuous part of the hybrid plant consists of a set of first order integrators

$$\dot{x} = c_i \in \mathbb{R}^2, \quad i = 1, 2, 3, 4.$$

$$C = [c_1, c_2, c_3, c_4] = \begin{bmatrix} 0.5 & -1 & 1 & -1 \\ 1 & 1 & 1 & -0.4 \end{bmatrix}$$

and we associate the differential equation $\dot{x} = c_i$ with the transition t_i of the Petri net in Fig. 1. The control objective is to drive the state from the initial condition $x_0 = [1, -1]^T$ to the convex region $\mathcal{T}$ of the state space where

$$\mathcal{T} = \left\{ x \in \mathbb{R}^2 : \begin{bmatrix} 1 \\ 1 \end{bmatrix} \leq x \leq \begin{bmatrix} 1.1 \\ 1.1 \end{bmatrix} \right\}$$

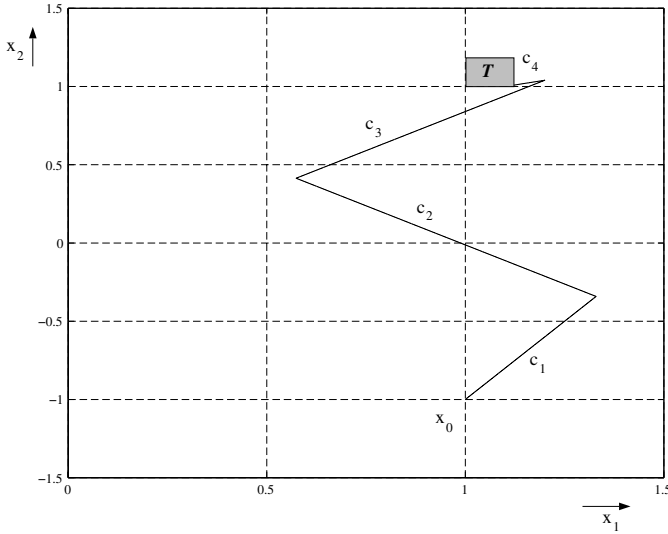


Fig. 5. The trajectory of the continuous state.

According to Proposition 3 we formulate the following linear programming problem

$$\begin{aligned} & \min (\tau_1 + \tau_2 + \tau_3 + \tau_4) \\ & \text{subject to : } \begin{cases} x_f = x_0 + \tau_1 c_1 + \tau_2 c_2 + \tau_3 c_3 + \tau_4 c_4 \in \mathcal{T} \\ \tau_i \geq \Delta \end{cases} \end{aligned}$$

where $\Delta = 0.1$. The solution of the linear programming problem gives

$$\tau_1 = 0.6585, \quad \tau_2 = 0.7554; \quad \tau_3 = 0.6261, \quad \tau_4 = 0.1$$

and we can drive the state from x_0 to $\mathcal{T}$ with one firing of each transition by setting $\sigma^{t_i} = \tau_i$, $i = 1, 2, 3, 4$. The trajectory of the continuous state is shown in Fig. 5.

6 Conclusions

In this paper, supervisory control of hybrid systems was addressed using a class of timed Petri nets named programmable timed Petri nets. New methodologies were introduced and algorithms were derived to address these issues. Sufficient conditions for supervisory control design were presented. For the case when the plant is a collection of affine systems or first order integrators with switching logic generated by a programmable timed Petri net, efficient algorithms for supervisory control synthesis were developed.

References

1. R.L. Grossman, A. Nerode, A.P. Ravn, and H. Rischel, editors. *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*. Springer-Verlag, 1993.
2. P. Antsaklis, W. Kohn, A. Nerode, and S. Shastri, editors. *Hybrid Systems II*, volume 999 of *Lecture Notes in Computer Science*. Springer, 1995.
3. R. Alur, T.A. Henzinger, and E.D. Sontag, editors. *Hybrid Systems III, Verification and Control*, volume 1066 of *Lecture Notes in Computer Science*. Springer, 1996.
4. P. Antsaklis, W. Kohn, A. Nerode, and S. Shastri, editors. *Hybrid Systems IV*, volume 1273 of *Lecture Notes in Computer Science*. Springer, 1997.
5. A. Stephen Morse, editor. *Control using logic-based switching*, volume 222 of *Lecture Notes in Control and Information Sciences*. Springer, 1997.
6. P.J. Antsaklis and A. Nerode, editors. *IEEE Transactions on Automatic Control, Special Issue on Hybrid Systems*, 1998.
7. P.J. Antsaklis and X.D. Koutsoukos. On hybrid control of complex systems: A survey. In *3rd International Conference ADMP'98, Automation of Mixed Processes: Dynamic Hybrid Systems*, pages 1–8, Reims, France, March 1998.
8. P. Peleties and R. DeCarlo. Analysis of hybrid systems using symbolic dynamics and Petri nets. *Automatica*, 30(9):1421–1427, 1994.
9. P.J. Ramadge. On the periodicity of symbolic observations of piecewise smooth discrete-time systems. *IEEE Transactions on Automatic Control*, 35(7):807–813, 1990.
10. J. Lunze, B. Nixdorf, and H. Richter. Hybrid modelling of continuous-variable systems with application to supervisory control. In *Proceedings of the European Control Conference 97*, Brussels, Belgium, July 1997.
11. J. Le Bail, H. Alla, and R. David. Hybrid Petri nets. In *Proceedings 1st European Control Conference*, Grenoble, France, July 1991.
12. S. Pettersson and B. Lennartson. Hybrid modelling focused on hybrid Petri nets. In *Proceedings 2nd European Workshop on Real-time and Hybrid systems*, pages 303–309, Grenoble, France, June 1995.
13. A. Giua and E. Usai. High-level hybrid Petri nets: A definition. In *Proceeding of the 35th Conference on Decision and Control*, Kobe, Japan, December 1996.
14. D. Vibert, C. Valentin-Roubinet, and E. Neil. A modelling method to take into account fluctuations of continuous variables in a class of hybrid systems. In *Proceedings of the European Control Conference 97*, Brussels, Belgium, July 1997.
15. J-M. Flaus and H. Alla. Structural analysis of hybrid systems modelled by hybrid flow nets. In *Proceedings of the European Control Conference 97*, Brussels, Belgium, July 1997.
16. I. Demongodin and N.T. Koussoulas. Differential Petri nets: Representing continuous systems in a discrete-event world. *Transactions on Automatic Control*, 43(4):573–579, 1998.
17. M.D. Lemmon, K.X. He, and C.J. Bett. Modeling hybrid control systems using programmable Petri nets. In *3rd International Conference ADMP'98, Automation of Mixed Processes: Dynamic Hybrid Systems*, pages 177–184, Reims, France, March 1998.
18. J.O. Moody. *Petri Net Supervisors for Discrete Event Systems*. PhD thesis, Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN, 1997.
19. N.H. McClamroch, C. Rui, I. Kolmanovsky, and M. Reyhanoglu. Hybrid closed loop systems: A nonlinear control perspective. In *Proceedings of the 36th IEEE Conference on Decision and Control*, pages 114–119, 1997.

20. John Guckenheimer. A robust hybrid stabilization strategy for equilibria. *IEEE Transactions on Automatic Control*, 40(2):321–326, 1995.
21. J.A. Stiver, P.J. Antsaklis, and M.D. Lemmon. Interface and controller design for hybrid control systems. In Panos Antsaklis, Wolf Kohn, Anil Nerode, and Shankar Sastry, editors, *Hybrid Systems II*, volume 999 of *Lecture Notes in Computer Science*, pages 462–492. Springer, 1995.
22. J.A. Stiver, P.J. Antsaklis, and M.D. Lemmon. An invariant based approach to the design of hybrid control systems. In *IFAC 13th Triennial World Congress*, volume J, pages 467–472, San Francisco, CA, 1996.
23. X.D. Koutsoukos and P.J. Antsaklis. An approach to hybrid systems control applied to clocks. In *Fifth International Hybrid Systems Workshop*, Notre Dame, USA, September 1997.
24. X.D. Koutsoukos, K.X. He, M.D. Lemmon, and P.J. Antsaklis. Timed petri nets in hybrid systems: Stability and supervisory control. *DEDS: Theory and Applications*, 8(2), 1998.
25. R. Alur and D. Dill. The theory of timed automata. *Theoretical and Computer Science*, 126:183–235, 1994.
26. R. Alur, T. Henzinger, and P-H Ho. Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering*, 22:181–201, 1996.
27. J.L. Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice-Hall, Englewood Cliffs, NJ, 1981.
28. W. Reisig. *Petri Nets*. Springer-Verlag, 1985.
29. T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of IEEE*, 77(4):541–580, 1989.
30. F. Baccelli, G. Cohen, and B. Gaujal. Recursive equations and basic properties of timed Petri nets. *DEDS: Theory and Applications*, 1(4):415–439, 1992.
31. K. Yamalidou and J.C. Kantor. Modeling and optimal control of discrete-event chemical processes using Petri nets. *Computers in Chemical Engineering*, 15(7):503–519, 1991.
32. J.O. Moody and P.J. Antsaklis. Supervisory control using computationally efficient linear techniques: A tutorial introduction. In *Proceedings of the 5th International IEEE Mediterranean Conference on Control and Systems*, Paphos, Cyprus, July 1997.
33. J.O. Moody and P.J. Antsaklis. Supervisory control of Petri nets with uncontrollable/unobservable transitions. In *Proceedings of the 35th Conference on Decision and Control*, pages 4433–4438, Kobe, Japan, December 1996.
34. Bernard Friedland. *Advanced Control System Design*. Prentice-Hall, 1996.
35. R. Alur, C. Courcoubetis, T.A. Henzinger, and P-H. Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In Robert L. Grossman, Anil Nerode, Anders P. Ravn, and Hans Rischel, editors, *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 209–229. Springer-Verlag, 1993.
36. Michael Tittus. *Control Synthesis for Batch Processes*. PhD thesis, Control Engineering Lab., Chalmers University of Technology, Göteborg, Sweden, 1995.
37. J.A. Stiver, P.J. Antsaklis, and M.D. Lemmon. An invariant based approach to the design of hybrid control systems containing clocks. In Rajeev Alur, Thomas A. Henzinger, and Eduardo D. Sontag, editors, *Hybrid Systems III, Verification and Control*, volume 1066 of *Lecture Notes in Computer Science*, pages 464–474. Springer, 1996.
38. B. Lennartson, M. Tittus, B. Egardt, and S. Pettersson. Hybrid systems in process control. *Control Systems Magazine*, 16(5):45–56, October 1996.

A Case Study in Tool-Aided Analysis of Discretely Controlled Continuous Systems: The Two Tanks Problem*

S. Kowalewski, O. Stursberg, M. Fritz, H. Graf, I. Hoffmann, J. Preußig, M. Remelhe, S. Simon, H. Treseler

Department of Chemical Engineering,
University of Dortmund, Germany.

Abstract. This case study compares the usefulness and applicability of eight computer tools with respect to the validation of logic control programs for continuous processes. Six simulation packages (Taylor's MATLAB-based simulator, SIMULINK/STATEFLOW, GPROMS, SHIFT, DYMOILA, and BASiP) and two verification tools (SMV and HYTECH) were applied to a single process control example with non-trivial continuous dynamics. The paper presents a detailed description of this benchmark example. Short introductions to the tools are given and the application results are described and discussed with emphasis on the suitability to the problem and the numerical performance.

1 Introduction

This contribution reports on a case study in the application of tool-based methods for a systematic analysis of hybrid systems. In particular, the study is focussed on continuous processes controlled by logic controllers, as this class of hybrid systems is widespread among process industry plants. To achieve comparable results, all methods and tools under consideration were applied to the same example. We chose a simple process from our laboratory for this purpose. It will be introduced in detail in the following section. This benchmark example differs from other examples in the literature (as, for example, the train-gate-controller [LS87] or the steam boiler problem [ABL96]) in the following respects:

1. The problem is formulated such that simulation tools as well as verification tools can be applied meaningfully and will find particular challenges. In the case of simulation, the user must identify the worst-cases properly when non-determinism comes into play, whereas the tool has to handle event-triggered

* This work is supported by the *German Research Council (DFG)* in the temporary graduate school *Modeling and Model-Based Design of Complex Technical Systems* (2nd, 4th and 8th author) and as a part of the special program *Continuous-Discrete Dynamics of Technical Systems (KONDISK)* under the grants En 152/19, En 152/22 and Ko 1430/3 (5th, 6th, 7th and 9th author). The third author is funded by the Volkswagen Foundation.

switching correctly. The main challenge in applying verification tools is to find a necessary abstraction of the continuous dynamics.

2. Although the plant looks quite simple (two connected water tanks, see next section) and the underlying physics are easy to understand, the continuous dynamics are far from being trivial (as can be seen by the Eqs. (5) – (8)). As a consequence, it is not obvious on first sight how a suitable abstraction can be made, if verification tools based on models with simpler dynamics (e. g. HYTECH with Linear Hybrid Automata) shall be applied.
3. The example is an existing plant and not a purely idealized model. This becomes obvious in the models of the valves' dynamics which were determined experimentally (Eqs. 7 and 8). Consequently, we are ensured that no unrealistic simplifications of the problem are introduced from the outset. Also, since the theoretical results can be checked at the real equipment, it is possible to investigate the effects of modeling errors or uncertainties.
4. The problem statement corresponds to an every-day situation in process automation engineering: An already developed control program has to be checked with respect to a non-formal, natural language requirement specification based on some knowledge about the process to control. Consequently, we use a common industrial representation for the control code in the example.

The paper is organized as follows. In Sec. 2, we describe the two tanks benchmark problem in detail. Section 3 is devoted to the simulation tools which are Taylor's MATLAB-based simulator [TK96], SIMULINK/STATEFLOW [Mat97], G-PROMS [PB94], SHIFT [DGS96], DYMOLA [Elm93] and BASiP [WFSE96]. Each tool is introduced shortly and the application results and user experiences are described. The numerical performance is compared with respect to the accuracy of state event detection and to computational efficiency. In Sec. 4, the suitability of the verification tools SMV [CGL94] and HYTECH [HHW97] is evaluated. The paper ends with a discussion of the results.

2 The benchmark example

2.1 The plant

The system under consideration is a small laboratory plant which is used for an practical exercise in continuous control for chemical engineering students at the University of Dortmund. It has been used for research purposes in the past, e. g. as a benchmark for nonlinear control design methods [HE94] or as an illustration of modeling concepts for hybrid systems [HK96,SKHP97]. The plant consists of two connected cylindric tanks T_1 and T_2 which are situated on different levels (see Fig. 1). The diameter of T_1 is 12 cm, the one of T_2 is 5 cm. Both tanks are 1 m in height. The connecting pipe is attached to T_2 at a height of $H = 0.39$ m. The plant is operated with water. The incoming flow $\dot{V}$ controlled by the valve V_{Input} . Since it has only two positions, open or closed, the input flow is given by:

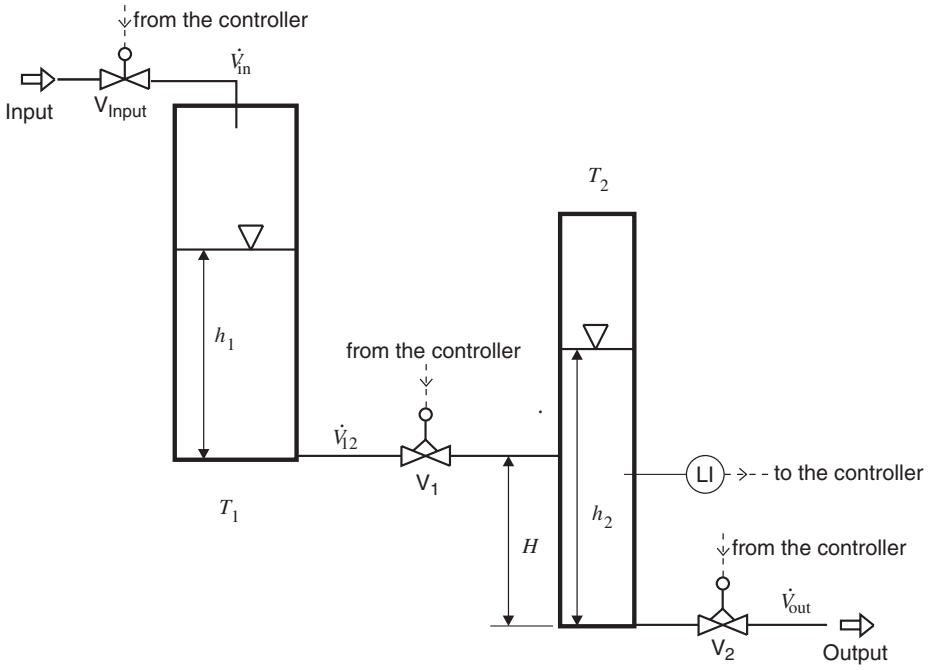


Fig. 1. Scheme of the laboratory plant

$$\dot{V}_{in} = \begin{cases} 0 \text{ l/h,} & \text{if } V_{input} \text{ closed,} \\ 400 \text{ l/h,} & \text{if } V_{input} \text{ open.} \end{cases} \quad (1)$$

The two other valves, V_1 and V_2 , are control valves which can take on any (normalized) position between 0 (completely open) and 80 (completely closed):

$$P_1, P_2 \in [0, 80]. \quad (2)$$

V_1 and V_2 are controlled by a binary signal: open valve and close valve. Opening and closing the valves is relatively slow: The complete process takes approximately 80 seconds. It can be assumed that the opening and closing speed is constant:

$$|\dot{P}_1|, |\dot{P}_2| = 1/s = \text{const.} \quad (3)$$

The dynamics of the two levels h_1 and h_2 can be derived from the mass balances for both tanks (A_1 and A_2 are the base areas of T_1 or T_2 , resp.):

$$\dot{h}_1 = \frac{(\dot{V}_{in} - \dot{V}_{12})}{A_1}, \dot{h}_2 = \frac{(\dot{V}_{12} - \dot{V}_{out})}{A_2}. \quad (4)$$

Applying Torricelli's law yields the following model for the flows $\dot{V}_{12}$ and $\dot{V}_{out}$ coming out of the two tanks. Note from Eq. 5 that the autonomous system already shows hybrid dynamics as the valid model for $\dot{V}_{12}$ has to be switched from one mode to the other when h_2 crosses the threshold H .

$$\dot{V}_{12} = \begin{cases} K_1(P_1) \cdot \sqrt{h_1 - (h_2 - H)}, & \text{if } h_2 > H, \\ K_1(P_1) \cdot \sqrt{h_1}, & \text{if } h_2 \leq H \end{cases} \quad (5)$$

$$\dot{V}_{out} = K_2(P_2) \cdot \sqrt{h_2}. \quad (6)$$

The coefficients K_1 and K_2 in Eq. 5 and Eq. 6 are necessary to represent the dependencies between the flows and the valve positions. The corresponding relations were determined experimentally:

$$K_1(P_1) = \begin{cases} 1,85 \cdot 10^{-4} \cdot e^{-6 \cdot 10^6 \cdot P_1^3} \frac{m^{5/2}}{s}, & \text{if } 0 \leq P_1 < 80, \\ 0 \frac{m^{5/2}}{s}, & \text{if } P_1 = 80. \end{cases} \quad (7)$$

$$K_2(P_2) = \begin{cases} 2,26 \cdot 10^{-4} \cdot e^{-5,7 \cdot 10^6 \cdot P_2^3} \frac{m^{5/2}}{s}, & \text{if } 0 \leq P_2 < 80, \\ 0 \frac{m^{5/2}}{s}, & \text{if } P_2 = 80. \end{cases} \quad (8)$$

2.2 The controller

Fig. 2 and 3 show the control program for the example process. The representation is in accordance with the international standard IEC 1131-3 for Programmable Logic Controllers [Int93]. The program is divided into two parts: the declaration part (Fig. 2) and the program body (Fig. 3). For the program body the language *Sequential Function Chart (SFC)* is chosen because it is particularly suited for sequence control applications.

The program does the following: After the start command the process is in “start-up-phase” consisting of the steps 1 and 2. During step 1, T_1 is filled by opening V_{Input} (S stands for set) while V_1 is kept closed. After the period **Time1** the transition to step 2 takes place (the variable **<stepname>.T** can be used according to IEC 1131-3 to determine the current duration of an active step). When step 2 becomes active, V_1 will be opened. After **Time2** the “start-up phase” is finished and the “stationary operation” begins in which the process is either in step 3 or step 4. The difference between these two steps is that in step 3 the output valve V_2 is opened whereas in step 4 it is closed (R stands for reset). Step 3 is active as long as the level in T_2 remains above the lower threshold **L_minus**. As soon as it drops below, the controller moves to step 4 until the level rises above the upper limit **L_plus**.


```

PROGRAM TwoTanks

VAR_INPUT
  Start : BOOL;  (* Input from environment: start process *)
  Level : REAL(0..1000);  (* Level in tank 2 *)
END_VAR

VAR_OUTPUT
  V_Input : BOOL := FALSE;  (* Output: open/close input valve *)
  V_1 : BOOL := FALSE;  (* Output: open/close connection valve *)
  V_2 : BOOL := FALSE;  (* Output: open/close output valve *)
  (* for all valves: TRUE = open, FALSE = close *)
END_VAR

VAR CONSTANT
  Time1 : TIME := T#90s;  (* duration of step 1 *)
  Time2 : TIME := T#20s;  (* duration of step 2 *)
  L_plus: REAL(0..1000) := 940;  (* upper limit for level tank 2 *)
  L_minus: REAL(0..1000) := 160;  (* lower limit for level tank 2 *)
END_VAR

```

Fig. 2. Control program for the example: declaration part

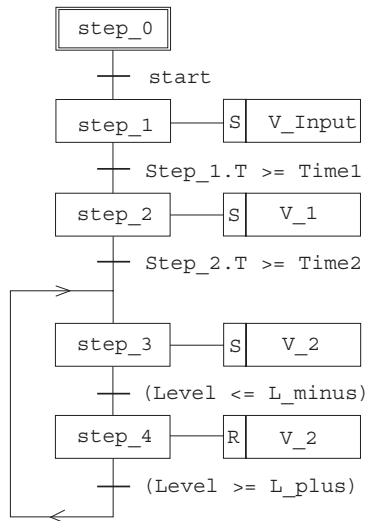


Fig. 3. Control program for the example: program body (SFC)

2.3 The requirements

The desired behavior of the two tanks system is specified by the following two requirements:

1. No overflow may occur.
2. The process must reach an equilibrium state in finite time. This means, the level in T_2 must not swing periodically between **L_minus** und **L_plus** in the stationary operation mode. Consequently, the control sequence has to stay in step 3 eventually.

2.4 The different cases

The requirements given in the previous section have to be checked for five different sets of control parameters (see Table 1). These values were chosen to capture most of the interesting scenarios which can take place in the process: 1. overflow of T_1 , 2. overflow of T_2 , 3. equilibrium state, 4. periodic behavior. The fifth set of parameters leads very closely to case 2 but does not violate the requirement. It is especially suited to determine the necessary grade of computational accuracy of the tools.

	Time1	Time2	L_plus	L_minus
1	T#90s	T#20s	940	160
2	T#70s	T#30s	940	160
3	T#70s	T#20s	940	160
4	T#60s	T#25s	900	300
5	T#70s	T#26.85s	940	160

Table 1. Different sets of control parameters for the program TwoTanks

In the first step of the analysis, it is assumed that no disturbances arise during the plant operation. Obviously, in this case the problem is purely deterministic and can be solved by proper simulation. To provide a problem for which the verification tools can employ their particular strengths in handling nondeterminism, the following scenario can be considered in a second step: A leakage may appear in tank T_1 or in the pipe between T_1 and V_1 . If this happens, approximately 10% of $\hat{V}_{12}$ will be lost. It can be assumed that the leakage will be discovered and closed within 2 minutes and that it will not appear again for the next 20 minutes after repairing.

3 Application of the simulation tools

3.1 Taylor's Matlab-based simulator

As a part of a project to define a modeling language for a broad class of hybrid systems, a hybrid simulation facility for the widely used modeling and simulation

environment MATLAB has been developed by Taylor [TK96]. It facilitates the detection and handling of state events in continuous time systems and thus supports jumps in the system trajectory as well as switchings of the model structure.

The evolution of the continuous state x is characterized by a set of differential equations $\dot{x} = f(x, m, t)$ depending on the mode input m . State events are detected by zero-crossings of a flag variable ϕ involving the state, time and the mode. Each integration point is calculated as a *trial* point according to the currently valid set of differential equations first, and is only added to the trajectory if no change of sign is detected. Otherwise, an iterative procedure is initialized to find a step-size h such that the flag variable equals zero (within a specified accuracy). Before the integration proceeds, the model state may be reset, if necessary, and the new mode is calculated. Thus, contrary to standard integration routines, it is ensured that mode changes will only take place at the end of an integration step.

The rigorous handling of state events requires an extended MATLAB function block (Fig. 4) to determine, at each integration step and during the iteration procedure, not only the derivative of the continuous state but also the flag variable ϕ . In addition, if a state reset becomes necessary, a new value r for the continuous state variable x is calculated by a supplementary function h . As an additional input, the mode value m influences the calculations. Since it is computed as the sign of the flag variable ϕ , it is restricted to the values 1, -1 or 0.

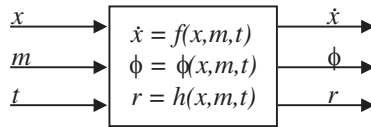


Fig. 4. Extended MATLAB function block

Modeling the two tanks system was realized by defining such a function block which describes the evolution of the position of the valves and the liquid levels depending on the related mode. In case of the valves three different modes have to be distinguished, corresponding to opening, stagnating and closing. The related flag variables depend on whether a critical level is reached and on the actual valve-position. To consider the coupling/decoupling of the tanks' dynamics one more flag variable had to be defined depending on the level in tank two. The related mode only takes on the values 1 or -1. In this example, a reinitialisation of the state vector at switching points was not necessary. However, due to the way of the mode calculation, the flag variables which are related to the valve positions have to be reset to zero whenever a valve is completely closed/opened so that the corresponding mode switches to zero (stagnating). Since the integration algorithm currently does not support the handling of time events, an additional

script file has to be created which defines the sequence of integrations. At the beginning of each integration procedure, the flag variables are initialized within the function block according to a global variable that represents the current step in the control procedure. The system was simulated using an extended variable-step Runge-Kutta-Fehlberg integration routine. The simulation results can easily be visualized using the routines provided by MATLAB as can be seen in Fig. 5 for the parameter set 1. Note that h_1 is not limited to 1m in this model because we were only interested in determining the overflow of T_1 .

To summarize, the simulator could handle the benchmark problem correctly. The accurate handling of state events provided correct results for every set of parameters. The described restriction due to the calculation of the mode as the sign of a flag variable did not apply in this case because for each mode only two or three different values had to be distinguished. However, it complicates the definition of the flag variables and requires a comprehension of the underlying integration algorithm to ensure correct iteration results.

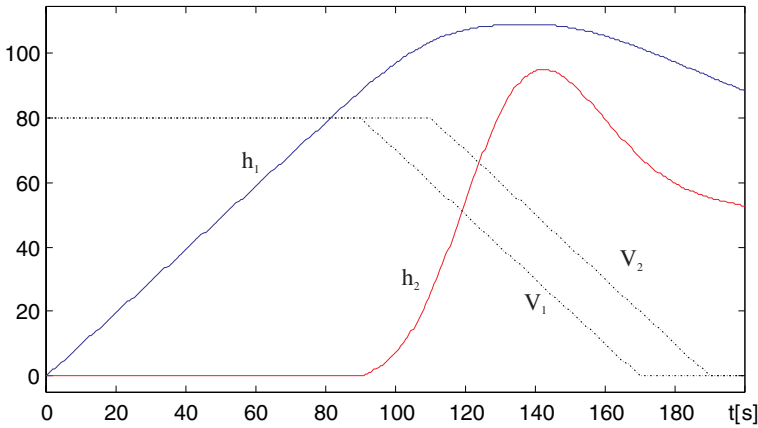


Fig. 5. Visualization of simulation results by MATLAB (parameter set 1)

3.2 Simulink/StateFlow

Another possibility to simulate hybrid systems in the MATLAB framework is provided by the SIMULINK software package. SIMULINK is a graphical, block-diagram oriented tool for modeling, simulating, and analyzing linear and nonlinear dynamic systems which is based on MATLAB's representations and numerical routines. The latest release of the package introduces a toolbox called STATE-FLOW which makes it possible to introduce discrete event systems as blocks in a SIMULINK block diagram. This new feature makes SIMULINK particularly inter-

esting for hybrid system simulation and was the motivation to include the tool in this comparison.

STATEFLOW uses statecharts [Har87] as the modeling paradigm to describe the behavior of a block. Technically, each STATEFLOW block is a masked SIMULINK model. This means, STATEFLOW transforms each statechart into the SIMULINK-specific internal representation, a so-called *S-function*. This S-function is the agent SIMULINK interacts with for simulation and analysis. The control behavior represented by the statechart complements the algorithmic behavior modeled by the standard block by switching variable values. Obviously, this makes it possible to switch continuous behaviors during a simulation run depending on state events.

Modeling the two tanks example is straightforward in SIMULINK. We first implemented the plant dynamics by introducing blocks for the mass balances of the two tanks and the characteristic curves of the valves. The autonomous switching of the model at $h_2 = 0.39$ m was realized by a 'switch' block in the feedback loop from h_1 and h_2 to the input of the two tank balances. The easiest part was to build the model of the control program. Since the given SFC is merely a sequential machine and does not include any parallel behavior, it can be transformed into an isomorphic statechart (see Fig. 6). Coupling this STATEFLOW block to the other blocks via the valve commands and the level h_2 gives us the desired hybrid model of the system.

Correct state event detection during the integration of switched continuous models can be enforced in SIMULINK by use of the so-called 'hit crossing' block. It forces the simulation to locate zero crossings of the input signal. The block detects the zeroes (with machine precision) by choosing a smaller step-size (in the neighborhood of the crossing point) for the integration progress. We employed two of these blocks for detecting the crossing of the thresholds `L_plus` and `L_minus`.

Care has to be taken when real-time constraints are specified in the statechart of a STATEFLOW block, e.g., activity durations as in our example. For the simulation, the time steps in STATEFLOW are different from the time steps used for the standard continuous SIMULINK blocks. To realize a proper synchronization, the variable *systemtime* has to be fed into the STATEFLOW block as an additional input.

In conclusion, the addition of STATEFLOW to SIMULINK provides a comfortable environment for modeling and simulating of hybrid dynamical systems. When applied to the two tanks example, the modeling step was straightforward whereas the simulation required some experience with respect to the synchronization of the statechart and the standard blocks.

3.3 gPROMS

The development of the gPROMS package (*general PROcess Modeling System*) [PB94] was initiated in 1988 at Imperial College, London, and is currently still ongoing. The tool can be regarded as a successor of the now commercialized package SPEEDUP which is the most used simulator in the processing industries.

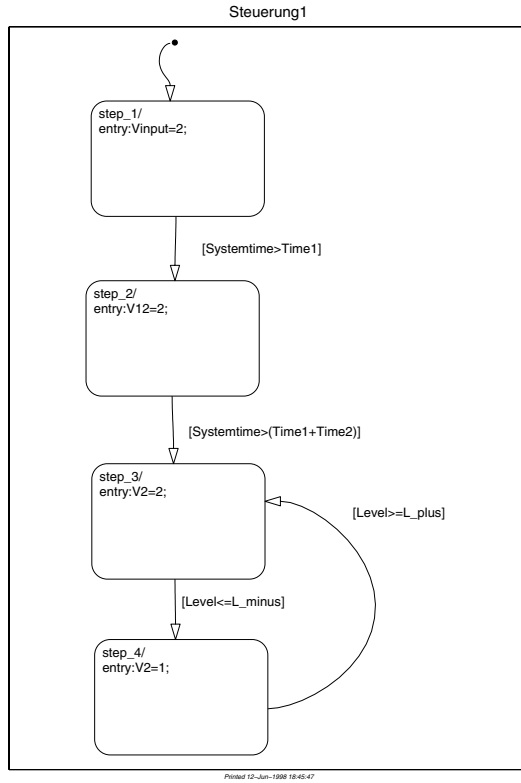


Fig. 6. Control program *TwoTanks* as a statechart in STATEFLOW

GPROMS also aims at process engineering applications and provides a general-purpose modeling environment for the entire range of processes from purely continuous to purely batch. In addition, capabilities for dynamic and steady-state simulation, optimization and visualization are included. Particular attention has been paid to handle modeling complexity by the concept of hierarchical submodel decomposition.

GPROMS distinguishes between two types of modeling entities: *models* and *tasks*. *Models* describe the physicochemical mechanisms governing a plant behavior formulated as integral, partial and ordinary differential, or algebraic equations, whereas *tasks* characterize the operating policies and control strategies imposed on the process. The *tasks* and *models* are combined to a *process* which delivers thus a description of the controlled system.

Modeling the two tanks plant, the mixed set of algebraic and differential equations as well as its switching discontinuities are defined in a *model*. Since the syntax of the GPROMS modeling language permits the use of *if-then-else*- as well as *case*-structures, the code for the description of the plant's behavior

can be comfortably generated. The operating procedure of the plant consisting of four different steps is expressed in four *tasks*. Each task is triggered by the switching of a global mode variable which observes the two tank levels. Both modeling entities form the whole *process* in which the sequential execution of the tasks is prescribed.

For the simulation, a relative accuracy of $1e-6$ has been set to restrict the local truncation error, and a monitoring step length of 1 second has been chosen. The simulation results have been visualized with the GRMS (*GPROMS Results Management System*) facility. A typical simulation result can be seen in Fig. 7. It shows the overflow of the second tank for parameter set 2. Again, no limits for the heights were necessary in the model.

However, if only simulation is used to treat problems with nondeterministic aspects, the right choice of the considered cases is left to the user's intuition. Dimitriadis et al. suggested an approach in which the worst case scenarios are detected automatically [SPD95]. For this purpose, the process model is augmented by binary variables assessing the systems safety in every time step of the system's possible trajectories. With the GPROMS optimization facility GOPT, a worst case of the system's behavior can then be found by a minimization or maximization (depending on the safety evaluation strategy) of the sum of all binary variables.

In general, the modeling of hybrid systems with GPROMS is comfortable. However, there are still only little diagnostic facilities to trace simulation failures back and solve them. The possibility of debugging and on-line variable monitoring is limited. As a conclusion, it can be said that GPROMS is well suited for the two tanks example, although GPROMS offers no facilities for a sound verification of the control program. The proof of fulfilment of the specified plant behavior requirements for the chosen sets of parameters could be given by identifying and simulating the corresponding worst-case situations.

3.4 Shift

SHIFT (Hybrid System Tool Interchange Format) is a specification language and simulation tool for hybrid systems, currently developed at Berkeley [DGS96]. The language allows an object-oriented modeling of multicomponent systems with hybrid behavior, i. e. '*components*' are instantiated from predefined classes, called '*types*', and features as the inheritance of properties are available. The overall model ('*world*') is build from the set of components which interact via input and output variables, and event synchronization. Though some modifications referring to the connection and synchronization of components have been introduced [DGV97], the theoretical basis for the description of the components' behavior is the *Hybrid Automata* paradigm defined by Alur et al. [ACH⁺95]. Hence, a component's continuous evolution is given by a DAE-System (the '*flow*') and the discrete dynamics by a finite state automaton, where event labels, guards, invariants and actions can be assigned to the transitions. The world behavior follows from the composition of all component's evolution.

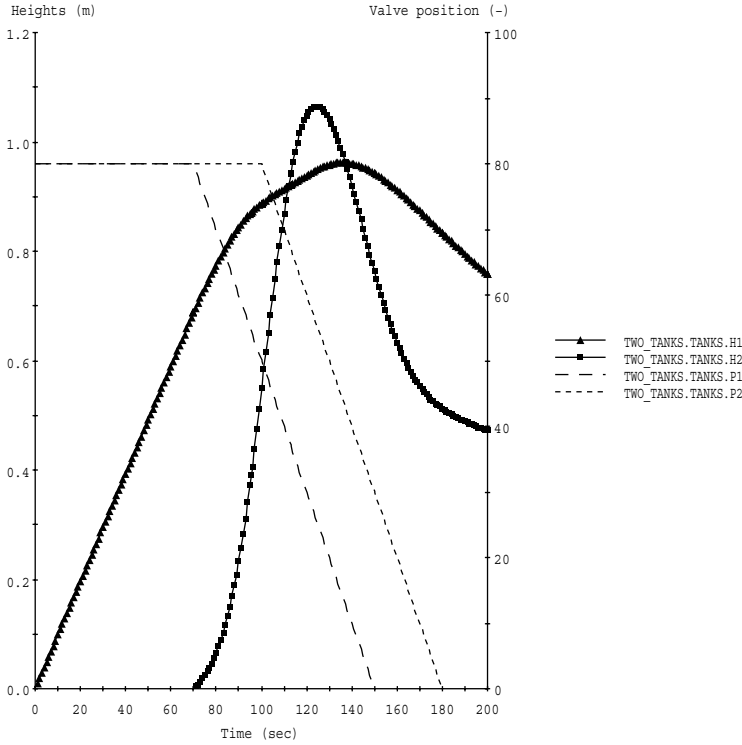


Fig. 7. Visualization of the simulation results by GRMS (parameter set 2)

In difference to the other tools used in this comparison, SHIFT enables the user to model and simulate dynamically reconfigurable hybrid systems: An action assigned to a discrete event of one component can trigger the creation of a new component or the rearrangement of the connection between components. This allows to describe networks which change their structure at run-time, as it appears for example when modeling automated traffic systems.

The SHIFT tool comprises a compiler to translate SHIFT specifications to C and to generate an executable simulation file. Furthermore, it offers an online-debugger and a data visualization and animation facility (TKSHIFT) to carry out simulation. Starting with an initialization phase, a simulation runs by the stepwise execution of continuous and discrete state evolutions in the order of their occurrence. Using the SHIFT version 1.8 for this investigation, only a 4th-order Runge-Kutta-method with fixed step-size was available to evaluate the continuous evolution. If the simulation generates several discrete events at one time instant, these are executed in a first-in-first-out strategy, which means, no user-specified interaction is possible in the case of non-deterministic discrete behavior.

To model the two tanks example, we created components for each of the system's technical devices (tanks, valves, connecting pipes) as well as for the supervising controller. The links between these components were constituted by defining the continuous quantities (liquid levels, flowrates, valve positions, etc.) as input/output variables and setting the connections correspondingly in a superordinated block. The components' discrete states are naturally given by discretizing the range of the continuous variables according to thresholds being important for the control objective (e. g., 'opening', 'steady', 'closing' for a control valve). A transition between two of these discrete states is introduced for the case that the continuous variable crosses the separating threshold in one direction. For the 'controller' component, a reset of the variables denoting the valves' mode ('opening', 'closing', 'steady') is assigned to the transitions as action. For components inhering continuous dynamics, appropriate algebraic and differential equations define the flow within the discrete states. Specifying the two tanks system in this manner, TkSHIFT produces plots as shown in Fig. 8 for the case of a periodic switching of the outlet valve (P1_n, P2_n denotes the normalized setting of the valves V_1 and V_2 : it equals 1 if closed and 0 if opened).

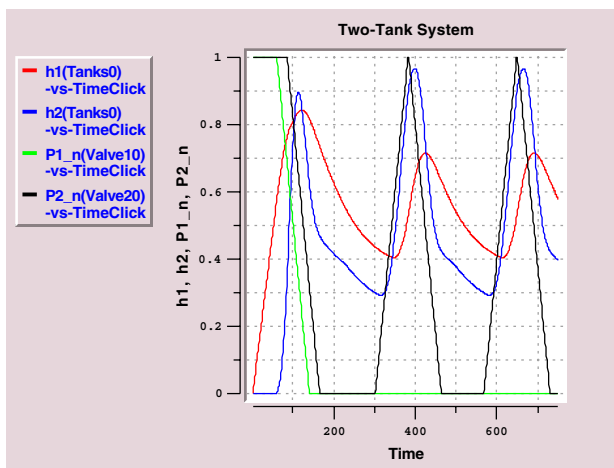


Fig. 8. The simulation results visualized by TkSHIFT (parameter set 4)

When applying SHIFT to the example with the different parameters and modes, we found that SHIFT is suitable to model processes of this type whereas some of its main features, as the dynamic reconfiguration or the inheritance of the components' properties, were not used. Some restrictions are given by the SHIFT-syntax, e. g. no if-then-else-structures are permitted in the flow definitions. Furthermore, the simulation algorithm does not offer the versatility of tools like GPROMS, DYMOLA or SIMULINK - e. g., no facility for an accurate detec-

tion of state events while calculating the continuous state evolution is available. Since SHIFT does not offer any verification procedures, tasks like analyzing the example control program in the disturbed case can only be solved by worst-case simulation.

It shall be mentioned that in newer versions of SHIFT some modifications to lessen or remove the disadvantages stated have been introduced: A variable step-size integration algorithm is available now, such that the integration precision should no longer considerably differ from that of the MATLAB tool or GPROMS. Additionally, in [GKV97] an interpolation method for a more accurate detection of state events is announced.

3.5 Dymola

DYMOLA is a general-purpose modeling and simulation environment. It was developed in 1978 at Lund Institute of Technology in Sweden [Elm93] and has been distributed commercially by Dynasim AB since 1993. The most relevant concepts realized in DYMOLA are object oriented modeling and non-causal modeling.

Object oriented modeling allows the independent definition of partial models, the construction of model libraries and the hierarchical model composition. These tasks are supported in DYMOLA by a graphical model editor and browser. A new model can be built by defining its equations and/or connecting already existing modules graphically.

Basic modules need not to be specified by any program language, instead the only thing to do is to declare the variables and parameters, to write down the equations, to define the interface vectors and to create an object symbol. Because of the object oriented concept, DYMOLA is suitable for multi domain modeling. Subsystems of different domains, for example electrical, mechanical and control systems, which are modeled with domain specific symbols, can be connected to an unique model.

Non-causal modeling means that it is possible to give the equations which describe the system behavior in any form. It is neither necessary to specify input or output variables nor to convert the equations to assignment statements. Even implicit equations can be specified. Therefore the interfaces of the submodel blocks are in general neither inputs nor outputs and feedbacks need not to be considered by additional connections.

When the model is being transformed into a set of equations, algebraic equations are inserted for each connection. This corresponds to Kirchhoff's current law. Symbolic equation processing is used to obtain the best possible assignment sequence, which makes the simulation more efficient. The DAE system is converted to state-space form if possible or to reduced DAE form. The non-causal modeling in combination with the symbolic equation processing enables both more simple models and more efficient simulation.

Discrete components can be modeled on basic language level by *if-then-else* expressions or by *when* statements. A limiter for example can be written as:

```

y = if u>Highlimit then Highlimit
    else if u<Lowlimit then Lowlimit
    else u

```

A when statement has the following general structure:

```

when <condition> then
    <assignments>
end when

```

DYMOLA translates these basic mathematical elements into time and state events when generating code. Modeling the two tanks example was realized by defining two subsystems: the controller and the physical system. The controller was modeled as a Petrinet (Fig. 9) using the components of the included Finite State Machines and Petrinet library which is built on basic discrete language elements.

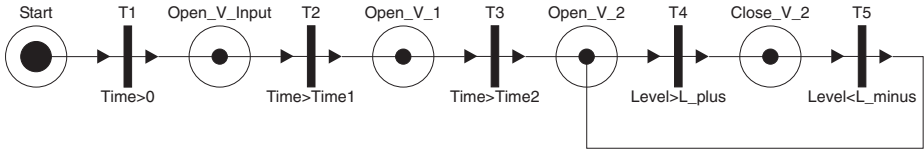


Fig. 9. Control program two tanks as a Petrinet in DYMOLA.

Additional Boolean expressions are necessary to derive the control signals to the valves from the states of the Petrinet. The physical system contains the equations from Section 2 without any alterations using *if-then-else* expressions. Only Eq. 3 needed to be modified to

```

der(P1) = if P1<0 and V_1 then 0
          else if P1>80 and not V_1 then 0
          else if V_1 then -1
          else +1

```

To summarize, DYMOLA is well suited to treat the two tanks example. However, it cannot take advantage of its symbolic equation processing because the example is not complex enough.

3.6 BASiP

As opposed to the other tools in this comparison, BASiP (Batch Simulation Package) is not a general-purpose modelling and simulation tool, but designed for a specific application domain: recipe driven chemical batch processes [WFSE96]. Though this example is not exactly what is generally understood as a batch

process, with some modelling tricks and some extensions to the program it is possible to model and simulate this system.

The key idea of BASiP is to provide a flexible and open software architecture that allows to integrate different components for modelling, simulation, and visualization [FE97].

By now, BASiP consists of a graphical user interface to model plant and recipes separately, a choice of different simulators, and several interfaces to visualization programs (e. g. GNUplot, see Fig. 10). An animation of the simulation run is also possible using the graphical editors. In a configuration shell, simulation experiments can be configured by simply putting together the desired models and components.

On the simulator side, two different approaches to simulate hybrid systems (of which batch processes are a typical example) are integrated. The first uses a system of ODEs that can be switched. Special attention has been given to event detection algorithms and a dynamic reconfiguration of the state vector after each switching to minimize the number of currently active equations. Another, novel approach is based on the discrete-event simulation paradigm. Here, the process description is simplified using algebraic equations that are evaluated to predict the next event in the simulation.

If the process is viewed from a sufficiently high level, this approach can produce results with no loss of accuracy compared to continuous simulation but in orders of magnitude less computation time. For this example, however,

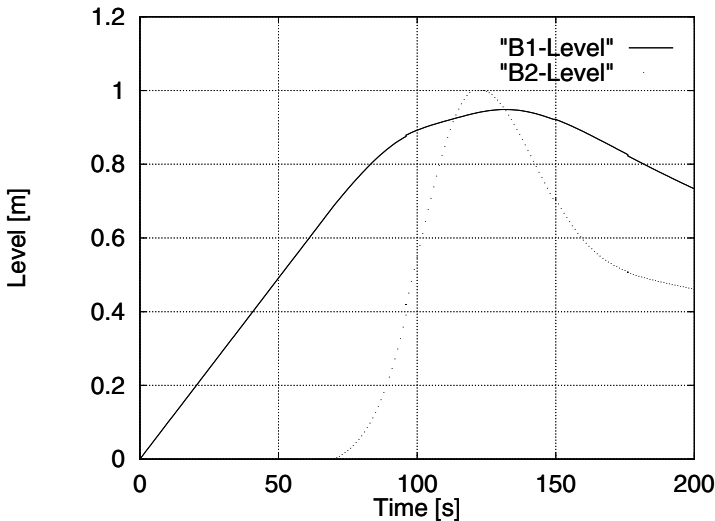


Fig. 10. The simulation results of BASiP visualized by GNUplot (parameter set 3)

the process is viewed on a level of detail that is actually above the scope of BASiP. Thus, with the simplifications taken in the discrete-event approach, no satisfactory results can be achieved. Only the continuous approach, extended by some new model blocks, can handle this system.

The sequence control of the example can easily be translated into a BASiP recipe, as both use the same representation as a sequential function chart. However, the semantics of a recipe are slightly different to that of sequence control because recipes lack the notion of state that can be set and reset. To represent the “automatic” flow operations between the tanks (flow takes place although no corresponding recipe phase is active), parallel branches with dummy filling operations were included in the recipe.

Because modelling in BASiP normally takes place on a higher level than required here, some model blocks had to be extended, while some new ones had to be added. This includes valves that have limited positioning speed or level-dependent flow operations. The disturbed case requires additional modifications of the models, and could not be implemented so far. Modifications or additions of model blocks require changes in the source code, written in C++, thus making such changes quite inconvenient. Future releases shall enable at least minor modifications without these efforts.

As a summary, it can be said, that BASiP proves to be apt even for cases it is not designed for, but lacks of flexibility to adapt the program to special cases.

3.7 Comparison of numerical results

In this section, we take a closer look at the detailed numerical results for the two tanks example provided by the different simulators. Since this case study investigates the suitability to hybrid systems, our special interest is in the handling of discrete phenomena. In particular, we look at how accurate state events, i.e. the switching point and the corresponding time, are determined. For this purpose, the fourth parameter set is chosen because in this case the state events “level h_2 reaches the threshold `L_plus` or `L_minus`, resp.,” occur periodically (cf. Fig. 7). This allows us to examine how the error can increase by accumulation during multiple occurrence of state events.

Table 2 shows the results of the investigation. It presents the computed times when h_2 reaches `L_minus` for the first (1st event) and the tenth time (10th event), together with the corresponding value of h_2 . The main conclusions which can be drawn from it are the following.

- It is reasonable to derive from the rows 3, 9 and 10, which represent the most accurate simulation runs of three of the tools with state event detection, that the correct values have to be near to 303.13 s for the first and 2669.9 s for the tenth event.
- As expected, the best results (relative error $< 1e-4$) are achieved when integration routines with step-size adaption, state event detection mechanisms and sufficiently small tolerances are used. However, the rows 1 and 2 show that also a fixed step-size can lead to satisfying results when event detection is employed.

- If neither variable step-sizes nor state event detection is available, the errors are of greater magnitude (cf. rows 4, 5, 12 and 13). In these cases, it is necessary to decrease the step-size which will increase the computational effort.
- The results of SIMULINK/STATEFLOW are worse than expected, in particular row 7. This may be due to the problem of initializing the interaction between the statechart for the controller and the Simulink model for the rest of the plant. We found no way to define the values of the exchange variables (opening/closing the valves) for the first integration step. As a workaround, the initial step was set to 1e-7 (which was only possible for the variable step size iteration). However, this does not explain the relatively large errors (magnitude 1e-3) in rows 6 and 7 and that the time value of the 10th event is worsened by use of the hit crossing blocks.

#	step-size	tolerance	1st event		2nd event	
			time [s]	value [m]	time [s]	value [m]
Matlab/Taylor (4th order Runge-Kutta)						
1	1	–	303.141	.300000	2670.5	.300000
2	0.1	–	303.133	.300000	2670.0	.300000
3	var, max 5 s	1e-6	303.127	.300000	2669.9	.300000
Simulink/StateFlow (Dorman-Prince)						
(a) Without hit crossing blocks						
4	1	–	307	.299260	2711	.299378
5	0.1	–	303.5	.299936	2671.4	.299998
6	var, max 1 s	rel. 1e-6, abs. 1e-15	303.76	.300000	2670.50	.300000
(b) With hit crossing blocks						
7	var, max 1 s	rel. 1e-6, abs. 1e-15	303.66	.300000	2666.14	.300000
gPROMS (Backward Differentiation Formula)						
8	var	rel. 1e-4, abs. 1e-4	303.21	.300000	2670.82	.300000
9	var	rel. 1e-6, abs. 1e-15	303.13	.300000	2669.87	.300000
Dymola (Adams/Bashforth/Moulton combined with Gear)						
10	var	1e-6 (rel. and abs.)	303.126	.300000	2669.88	.300000
BaSiP (Euler)						
11	var, max 1 s	1e-8	303.849	.300000	2673.66	.300000
Shift (4th order Runge-Kutta)						
12	1	–	301	.299464	2695	.299510
13	0.1	–	302.3	.299947	2666.1	.299944

Table 2. Numerical results of specific simulation runs

Of course, the values of Table 2 have to be interpreted with caution. It was impossible to provide absolute equivalent conditions for a comparison. The tools differ too much in many respects, e.g. available integration routines (the relatively weak results of BaSiP in row 11 are due to the fact that only an Euler

routine is implemented) and possibilities to adjust maximal and minimal step-sizes and tolerances. Additionally, the execution of the simulation programs is different. In some tools an interpreter processes the code, other tools generate C code which has to be compiled and executed externally. For this reason, CPU times are not included in Table 2. They were in the range of seconds for each of the tools except for Taylor's simulator which uses the MATLAB interpreter and therefore was much slower.

4 Application of the verification tools

4.1 HYTECH

HYTECH is a model checker for hybrid systems [HHW97]. To analyze systems with HYTECH they must be modeled as Linear Hybrid Automata [ACH⁺95]. For this example a subclass of hybrid automata, namely Timed Automata (TA) was chosen, although a system description with Linear Hybrid Automata would have been less complex and more accurate. Due to a known integer overflow problem in the analysis of hybrid automata, TA turned out to be much more suited in terms of feasible complexity. The model of the two tanks system is illustrated in Fig. 11. It consists of three TA, one describing the levels in the two tanks (**Levels**), one describing the position of the outlet valve (**Valve2**) and one for the behavior of the controller (**Controller**). The valve between the two tanks was introduced with a constant open position and a leak in the system was not considered. The analysis of the model can now be used to reveal whether or not the controller manages to avoid an overflow in $T2$ for given starting levels $L1$ and $L2$. In Fig.11 the model of the two tanks system is illustrated. The controller cycles between four locations, opening and closing $V2$ whenever the level in $T2$ hits the minimum/maximum landmark L_{minus}/L_{plus} . The TA **Valve2** changes its locations (which represent the discrete position of the valve) according to a clock which captures the dynamics of the valve. It propagates a change of the discrete state on to the TA **Levels** and reacts to open/close signals from the TA **Controller**. Each location in **Levels** represents a triple of discrete values for the levels in $T1$ and $T2$ and the position of $V2$. A clock captures the dynamics of the possible change of both levels according to the current levels and position of $V2$. To create a TA for the continuous behavior the given differential equations had to be discretized. This was done based on simple approximation method which maps the continuous state space into a set of discrete points that yield the locations of the discrete model. Then the successors of each location are evaluated by extending the vector of the state variable derivatives up to the next plane with discrete points. Then, all points directly adjacent to this intersection point are taken as the successors.

The size of such a discrete model depends on the number of discrete levels considered for the two tanks (N_{T1} , N_{T2}) and the number of discrete positions considered for the outlet valve (N_{V2}). To derive detailed models of the system a MATLAB function was implemented, which takes three parameters, namely the numbers of discrete points given by N_{T1} , N_{T2} , and N_{V2} , and then generates

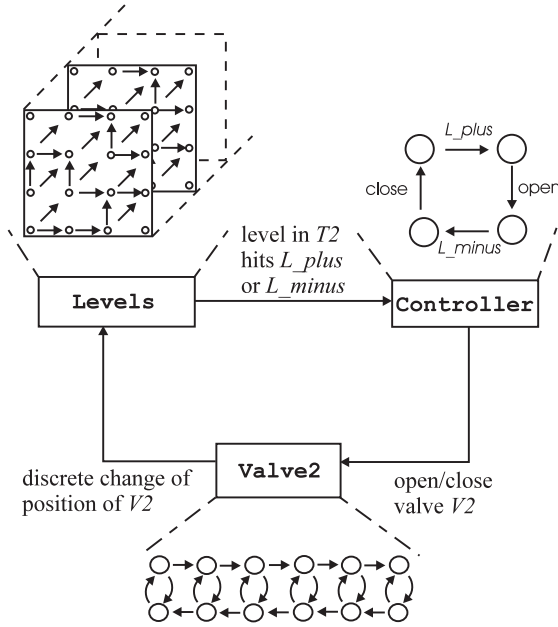


Fig. 11. Timed automata model of the Two Tank System

a HYTECH input file with the corresponding TAs. Another MATLAB function reads the HYTECH output and generates a graphical representation of the set of reachable locations. The most detailed model which HYTECH could still analyze on a Hypersparc with 100Mb RAM was derived with $N_{T1} = N_{T2} = 40$ and $N_{V2} = 20$. The resulting TA for the representation of the levels in the two tanks had more than 30.000 locations. The set of 2.100 reachable locations for this model from the initial levels $L1 = 0.7m$ and $L2 = 0.8m$ with landmarks at $L\_plus = 0.8m$ and $L\_minus = 0.2m$ is given in Fig. 12. The figure reveals that the level in Tank 2 does not exceed the 40th discretization point which corresponds to $1m$. The results show that HYTECH is in general suited for problems like the two tanks example, although it soon exhibits its complexity limits. Still, more accurate results could be achieved with less modeling effort, if HYTECH's analysis algorithms performed better on linear hybrid systems.

4.2 SMV

SMV (Symbolic Model Verifier) is a tool for applying the verification technique Symbolic Model Checking [CGL94]. The SMV-System provides a language which allows to describe the considered transition system hierarchically. The model checker uses an Ordered Binary Decision Diagrams (OBDD) based search algorithm to check the specification of the desired behavior given as a Computation Tree Logic (CTL) formula. If the system does not show the desired behavior, SMV produces an execution path with a counter-example.

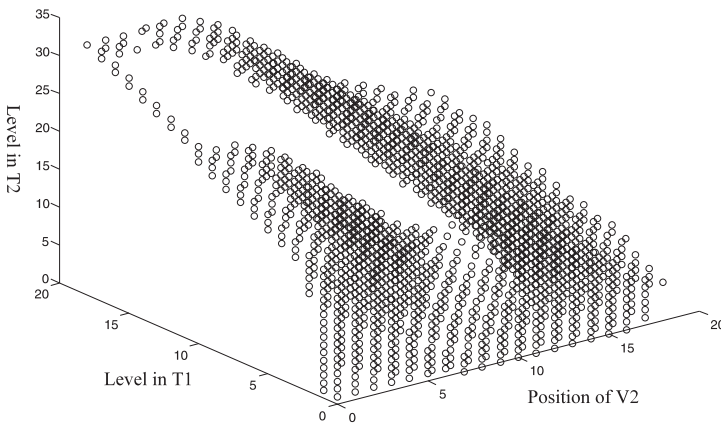


Fig. 12. Graphical representation of the reachability set determined by HyTECH

SMV has been applied to the verification of logic process controllers by Powers et al. [TPP97]. The two tanks example differs from these applications in that the continuous dynamics are much more complex here. Just as in the HyTECH case, a discrete approximation has to be derived from the continuous model to be able to verify the two tanks system with SMV. Again, this was implemented by means of MATLAB. Here, the approximation algorithm partitions the continuous state space into boxes and calculates the possible residence times in each box from the differential equations. All subsystems, like the valves, the controller and the discretized liquid levels are modeled separately.

In contrast to HyTECH, SMV does not offer the possibility to model quantitative time information. Therefore the time had to be introduced as additional automata representing clocks which take one transition at each SMV execution cycle. Since SMV does not provide a synchronisation concept in the sense that transitions can be forced to be executed simultaneously, we keep an image of the discrete state space with the actual state always one time step behind. By this trick, events can be detected by evaluating conditions on the states in two systems.

The size of the SMV input file generated by MATLAB, of course, grows with the discretization grade. Since SMV imposes a restriction on the input file size, it could be applied to the two tanks problem only for very rough discretizations. As a consequence, fine points of the analysis, like the behavior for the fifth parameter set, could not be determined correctly.

5 Discussion

We posed a benchmark problem to verify requirements for a simple but real plant controlled by a logic control program and applied eight tools for simulation or verification of hybrid systems to it. To summarize shortly, it can be stated that due to the relatively complex continuous dynamics the two verification tools, HYTECH and SMV, had major problems. Although automatic abstraction techniques were developed successfully, both tools soon reached their complexity limits (SMV much earlier than HYTECH due to the additional discretization of time). On the other hand, all six simulation tools (Taylor/MATLAB, SIMULINK/STATEFLOW, GPROMS, SHIFT, DYMOLA, and BASiP) could solve the problem satisfactorily, but the user has to rely on his/her intuition to identify the worst cases when it comes to nondeterministic occurrences of disturbances. To discriminate further, SHIFT lacks of an exact identification of state events. Taylor's MATLAB toolbox and BASiP could only be applied after some programming for which deeper knowledge of the tool was necessary. SIMULINK with STATEFLOW probably is the most comfortable environment for the user in this comparison but it revealed accuracy problems. For GPROMS and DYMOLA the two tanks example is an easy task to solve. Both tools could not show their particular strengths.

Of course, our choice of tools for this case study is incomplete. Many simulation packages are missing. With respect to verification, not only other tools can be applied, it is also possible to use different methods for discretization of the continuous dynamics. We hope that our contribution will motivate other researchers to apply their tools and methods of interest to this example and, by doing that, make the comparison more comprehensive.

Acknowledgements

We thank *Sebastian Engell* for helpful discussions, *Mario Stobbe* and *Volker Roßmann* for their efforts during installation of SHIFT and STATEFLOW, resp., *Markus Weber* for his contributions to the SIMULINK/STATEFLOW model of the two tanks example, and *André Deparade* for compiling the L^AT_EX 2_ε version of this paper.

References

- ABL96. J.-R. Abrial, E. Börger, and H. Langmaack. *Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control*. LNCS 1165. Springer, 1996.
- ACH⁺95. R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.
- CGL94. E.M. Clarke, O. Grumberg, and D.E. Long. Model Checking and Abstraction. *ACM Trans. on Programming Languages and Systems*, 16:1512–1542, 1994.

- DGS96. A. Deshpande, A. Göllü, and L. Semenzato. *The SHIFT Programming Language and Run-time System for Dynamic Networks of Hybrid Automata*. PATH-Report, <http://www.path.berkeley.edu/shift/>, 1996.
- DGV97. A. Deshpande, A. Göllü, and P. Varaiya. *Hybrid Systems IV*, chapter SHIFT - A Formalism and a Programming Language for Dynamic Networks of Hybrid Automata, pages 113–133. LNCS 1273. Springer, 1997.
- Elm93. H. Elmquist. Object-Oriented Modeling and Automatic Formula Manipulation in Dymola. *SIMS'93, Scandinavian Simulation Society*, 1993.
- FE97. M. Fritz and S. Engell. An open software architecture for batch process simulation. *Comp. Chem. Eng.*, 21, Suppl. (Proc. PSE '97/ESCAPE-7):769–773, 1997.
- GKV97. A. Göllü, M. Kourjanski, and P. Varaiya. The SHIFT simulation framework: Language, model and implementation. In *Proc. 5th Int. Hybrid Systems Workshop, Notre Dame, Indiana*, September 1997.
- Har87. D. Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8(3):231–274, 1987.
- HE94. T. Heckenthaler and S. Engell. Approximately time-optimal control of a two-tank system. *IEEE Control Systems Magazine*, 14(3), 1994.
- HHW97. T.A. Henzinger, P.H. Ho, and H. Wong-Toi. HyTECH: A model checker for hybrid systems. *Software Tools for Technology Transfer*, 1(1,2):110–122, 1997.
- HK96. I. Hoffmann and K.-U. Klatt. *Hybrid Systems III*, chapter Modeling Hybrid Dynamical Systems. LNCS 1066. Springer, 1996.
- Int93. International Electrotechnical Commission, Geneva. *International Standard 1131: Programmable Logic Controllers, Part 3: Programming Languages*, 1993.
- LS87. N. Leveson and J. L. Stolzy. Safety analysis using petri nets. *IEEE Transactions on Software Engineering*, 13(3):386–397, 1987.
- Mat97. MathWorks. *Matlab User Guide*. Natwick, USA, 1997.
- PB94. C.C. Pantelides and P.I. Barton. Modeling of combined discrete/continuous processes. *AIChE Journal*, 40:966–979, 1994.
- SKHP97. O. Stursberg, S. Kowalewski, I. Hoffmann, and J. Preußig. *Hybrid System IV*, chapter Comparing Timed and Hybrid Automata as Approximations of Continuous Systems, pages 361–377. LNCS 1273. Springer, 1997.
- SPD95. N. Shah, C.C. Pantelides, and V.D. Dimitriadis. Modeling and safety verification of discrete/continuous processing systems using discrete time domain models. *Workshop on analysis and design of event-driven operations in process systems*, April 1995.
- TK96. J. H. Taylor and D. Kebede. Modeling and simulation of hybrid systems in Matlab. In *Proc. IFAC World Congress, San Francisco, USA*, volume J, pages 275–280. IFAC, July 96.
- TPP97. A. L. Turk, S. T. Probst, and G. J. Powers. *Hybrid and Real-Time Systems*, chapter Verification of Real-Time Chemical Processing Systems, pages 259–272. LNCS 1201. Springer, 1997.
- WFSE96. K. Wöllhaf, M. Fritz, C. Schulz, and S. Engell. BaSiP – Batch Process Simulation With Dynamically Reconfigured Process Dynamics. *Comp. Chem. Eng.*, 20 (972), suppl. (Proc. ESCAPE-6):1281–1286, 1996.

Hybrid Systems with Finite Bisimulations^{*}

Gerardo Lafferriere¹, George J. Pappas², and Shankar Sastry²

¹ Department of Mathematical Sciences
Portland State University, Portland, OR 97207, USA
`gerardo@mth.pdx.edu`

² Department of Electrical Engineering and Computer Sciences
University of California at Berkeley, Berkeley, CA 94720, USA
`{gpappas,sastry}@eecs.berkeley.edu`

Abstract. The theory of formal verification is one of the main approaches to hybrid system analysis. Decidability questions for verification algorithms are obtained by constructing finite, reachability preserving quotient systems called *bisimulations*. In this paper, we use recent results from stratification theory, subanalytic sets, and model theory in order to extend the state-of-the-art results on the existence of bisimulations for certain classes of planar hybrid systems.

1 Introduction

Hybrid systems consist of finite state machines interacting with differential equations. Various modeling formalisms, analysis, design and control methodologies, as well as applications, can be found in [3,4,5,10,17]. Formal verification is the main computational approach for analyzing properties of hybrid systems. One of the most important verification problems for hybrid systems is the reachability problem which asks whether trajectories of the hybrid system can reach certain undesirable regions of the state space. Since hybrid systems have infinite state spaces, the decidability of verification algorithms is very important.

A uniform framework for tackling the decidability issue is provided by the notion of *bisimulation*. Bisimulations are quotients of the original hybrid system that are reachability preserving. Showing that an infinite state hybrid system has a finite state bisimulation is the first step in proving decidability. This approach has been successfully applied to timed automata [2], multirate automata [1], and initialized rectangular automata [19,12]. It should be mentioned that the notion of bisimulation is closely related to the various consistency notions for discrete and continuous systems [7,8,18].

Since the discrete dynamics are already finite, it is clear that decidability results for hybrid systems depend crucially on the success of obtaining finite bisimulations for continuous dynamics. The cases considered so far in the literature dealt with simple dynamics: $\dot{x} = 1$ for timed automata [2], $\dot{x} = a$ for

^{*} Research supported by the Army Research Office under grants DAAH 04-95-1-0588 and DAAH 04-96-1-0341.

multirate automata [1], $\dot{x} \in [a, b]$ for rectangular automata [19], and $A\dot{x} \leq b$ for linear hybrid automata [11]. In this paper, we extend the bisimulation methodology to hybrid systems with more general dynamics. We first present the standard bisimulation algorithm which, upon termination, provides the desired finite bismilarity quotient. In [15], we used purely geometric methods to show that the bisimulation algorithm terminates for a class of hybrid systems with planar linear dynamics. In this paper, we combine mathematical techniques from differential geometry and recent results in logic model theory in order to prove existence of finite bisimulations for various new classes of hybrid systems with planar continuous dynamics. This convergence of mathematical logic and differential geometry also provides a natural framework for extending the decidability frontier for more general classes of hybrid systems. Such extensions will require pushing the boundary of decidable theories in mathematical logic.

Abstracting a discrete graph from a hybrid system requires the analysis of trajectories of vector fields and their intersection properties relative to a given collection of sets. Considering hybrid systems with arbitrary dynamics and arbitrary state partitions would soon lead to pathological situations. *Subanalytic sets* [6,13,21] provide a rich class of sets which have many desirable local intersection properties with trajectories of *analytic* vector fields. Subanalytic sets can also be partitioned into smooth embedded submanifolds in a form suitable for constructing a bisimulation. Such partitions are called *stratifications*. Moreover, we show that relaxing the class of vector fields or sets in some naive ways leads to pathological situations. On the other hand, the concept of *o-minimal* theories in logic [24,25,26] identifies classes of sets with good intersection properties suitable for the global study of trajectories of vector fields. The combination of techniques from both fields highlights the kind of properties of sets that play a central role in obtaining discrete abstractions.

The outline of the paper is as follows: In Section 2 we review the notion of bisimulations of transitions systems. In Section 3 we define the class of hybrid systems under study and describe the main bisimulation algorithm. Section 4 presents some basic facts about stratification theory and subanalytic sets and relates them to the construction of bisimulations. In Section 5 we present recent results in model theory which are used in Section 6 in order to obtain classes of systems for which the bisimulation algorithm terminates. Section 7 contains conclusions and issues for further research.

2 Bisimulations of Transition Systems

A transition system $T = (Q, \Sigma, \rightarrow, Q_O, Q_F)$ consists of a (not necessarily finite) set Q of states, an alphabet Σ of events, a transition relation $\rightarrow \subseteq Q \times \Sigma \times Q$, a set $Q_O \subseteq Q$ of initial states, and a set $Q_F \subseteq Q$ of final states. A transition $(q_1, \sigma, q_2) \in \rightarrow$ is denoted as $q_1 \xrightarrow{\sigma} q_2$. The transition system is finite if the cardinality of Q is finite and it is infinite otherwise. A region is a subset $P \subseteq Q$. Given $\sigma \in \Sigma$ we define the predecessor $Pre_\sigma(P)$ of a region P as

$$Pre_\sigma(P) = \{q \in Q \mid \exists p \in P \text{ and } q \xrightarrow{\sigma} p\} \quad (1)$$

Given an equivalence relation $\sim \subseteq Q \times Q$, we define a quotient transition system $t/\sim$ as follows: Let $Q/\sim$ denote the quotient space. For a region P we denote by $P/\sim$ the collection of all equivalence classes which intersect P . The transition relation $\rightarrow_\sim$ on the quotient space is defined as follows: for $Q_1, Q_2 \in Q/\sim$, $Q_1 \xrightarrow{\sigma}_\sim Q_2$ iff there exist $q_1 \in Q_1$ and $q_2 \in Q_2$ such that $q_1 \xrightarrow{\sigma} q_2$. The quotient transition system is then $T/\sim = (Q/\sim, \Sigma, \rightarrow_\sim, Q_0/\sim, Q_F/\sim)$.

Given a partition $\sim$ on Q , we call a set a $\sim$ -block if it is a union of equivalence classes. The partition $\sim$ is a *bisimulation* of T iff Q_O, Q_F are $\sim$ -blocks and for all $\sigma \in \Sigma$ and all $\sim$ -blocks P , the region $Pre_\sigma(P)$ is a $\sim$ -block. In this case the systems T and $T/\sim$ are called *bisimilar*. A bisimulation is called finite if it has a finite number of equivalence classes. Bisimulations are very important because checking reachability on the bisimilar transition system is equivalent to checking reachability of the original transition system [11]. Therefore, if T is infinite and $T/\sim$ is a finite bisimulation, then verification algorithms for infinite systems are guaranteed to terminate. If $\sim$ is a bisimulation, it can be easily shown that if $p \sim q$ then

- B1** $p \in Q_F$ iff $q \in Q_F$, and $p \in Q_O$ iff $q \in Q_O$
B2 if $p \xrightarrow{\sigma} p'$ then there exists q' such that $q \xrightarrow{\sigma} q'$ and $p' \sim q'$

Based on the above characterization, given a transition system T , the following algorithm computes increasingly finer partitions of the state space Q .

Algorithm 1 (Bisimulation Algorithm for Transition Systems)

Set $Q/\sim = \{Q_O \cap Q_F, Q_O \setminus Q_F, Q_F \setminus Q_O, Q \setminus (Q_O \cup Q_F)\}$
while $\exists P, P' \in Q/\sim$ and $\sigma \in \Sigma$ such that $\emptyset \neq P \cap Pre_\sigma(P') \neq P$
 set $P_1 = P \cap Pre_\sigma(P'), P_2 = P \setminus Pre_\sigma(P')$
 refine $Q/\sim = (Q/\sim \setminus \{P\}) \cup \{P_1, P_2\}$
end while

If the algorithm terminates, then the resulting quotient transition system is a finite bisimulation. The goal of the next sections is to show that the above algorithm terminates for transition systems generated by a class of planar hybrid systems.

3 Bisimulations of Hybrid Systems

In this paper, we focus on transition systems generated by the following class of hybrid systems.

Definition 1. A hybrid system $H = (X, X_0, X_F, F, E, I, G, R)$ consists of

- $X = X_D \times X_C$ is the state space with $X_D = \{q_1, \dots, q_n\}$ and X_C an analytic manifold.
- $X_0 \subseteq X$ is the set of initial states
- $X_F \subseteq X$ is the set of final states
- $F : X \rightarrow TX_C$ assigns to each discrete state $q \in X_D$ an analytic vector field $F(q, \cdot)$

- $E \subseteq X_D \times X_D$ is the set of discrete transitions
- $I : X_D \longrightarrow 2^{X_C}$ assigns to each discrete state a set $I(q) \subseteq X_C$ called the invariant.
- $G : E \longrightarrow X_D \times 2^{X_C}$ assigns to $e = (q_1, q_2) \in E$ a guard of the form $\{q_1\} \times U$, $U \subseteq I(q_1)$.
- $R : E \longrightarrow X_D \times 2^{X_C}$ assigns to $e = (q_1, q_2) \in E$ a reset of the form $\{q_2\} \times V$, $V \subseteq I(q_2)$.

Trajectories of H start at any $(q, x) \in X_0$ and consist of continuous evolutions or discrete jumps. Continuous trajectories evolve according to the continuous flow $F(q, \cdot)$ as long as the state remains in the invariant set $I(q)$. If the flow exits $I(q)$, then a discrete transition is *forced*. If, during the continuous evolution, the state (q, x) enters a region $G(e)$ for some $e \in E$, then discrete transition e is *enabled*, and the state may instantaneously jump from (q, x) to any $(q', x') \in R(e)$. Then the system evolves according to $F(q', \cdot)$. Notice that the discrete transitions allowed in our model are similar to the type allowed in initialized rectangular automata [19]. Finally, we assume that our hybrid system model is *non-blocking*, that is from every state either a continuous evolution or a discrete transition is possible.

Every hybrid system $H = (X, X_0, X_F, F, E, I, G, R)$ generates a transition system $T = (Q, \Sigma, \rightarrow, Q_O, Q_F)$ by setting $Q = X$, $Q_0 = X_0$, $Q_F = X_F$, $\Sigma = E \cup \{\tau\}$, and $\rightarrow = (\cup_{e \in E} \xrightarrow{e}) \cup \xrightarrow{\tau}$ where

Discrete Transitions $(q, x) \xrightarrow{e} (q', x')$ for $e \in E$ iff $(q, x) \in G(e)$ and $(q', x') \in R(e)$

Continuous Transitions $(q_1, x_1) \xrightarrow{\tau} (q_2, x_2)$ iff $q_1 = q_2$ and there exists $\delta \geq 0$ and a curve $x : [0, \delta] \longrightarrow M$ with $x(0) = x_1$, $x(\delta) = x_2$ and for all $t \in [0, \delta]$ it satisfies $x' = F(q_1, x(t))$ and $x(t) \in I(q_1)$.

The continuous τ transitions are time-abstract transitions, in the sense that the time it takes to reach one state from another is ignored. Having defined the continuous and discrete transitions $\xrightarrow{\tau}$ and $\xrightarrow{e}$ allows us to formally define $Pre_\tau(P)$ and $Pre_e(P)$ for $e \in E$ and any region $P \subseteq X$ using (1). Furthermore, the discrete transitions allowed in our hybrid system model result in

$$Pre_e(P) = \begin{cases} \emptyset & \text{if } P \cap R(e) = \emptyset \\ G(e) & \text{if } P \cap R(e) \neq \emptyset \end{cases} \quad (2)$$

for all discrete transitions $e \in E$ and regions P . Therefore, if the sets $R(e)$ and $G(e)$ are blocks of any partition of the state space, then no partition refinement is necessary in the bisimulation algorithm due to any discrete transitions $e \in E$. This fact, decouples the continuous and discrete components of the hybrid system as long as the initial partition in the bisimulation algorithm contain the invariants, guards and reset sets, in addition to the initial and final sets. This allows us to carry out the bisimulation algorithm independently for each discrete state.

More precisely, define for any region $P \subseteq X$ and $q \in X_D$ the set $P_q = \{x \in X_C : (q, x) \in P\}$. For each discrete state $q \in X_D$ consider the finite collection of sets

$$\mathcal{A}_q = \{I(q), (X_0)_q, (X_F)_q\} \cup \{G(e)_q, R(e)_q : e \in E\} \quad (3)$$

which describes all the relevant sets associated with discrete state q . Let $\mathcal{S}_q$ be the coarsest partition of X_C compatible with the collection $\mathcal{A}_q$ (by compatible we mean that each set in $\mathcal{A}_q$ is a union of sets in $\mathcal{S}_q$). The (finite) partition $\mathcal{S}_q$ can be easily computed by successively finding the intersections between each of the sets in $\mathcal{A}_q$ and their complements. These collections $\mathcal{S}_q$ will be the starting partitions of the following bisimulation algorithm for hybrid systems.

Algorithm 2 (Bisimulation Algorithm for Hybrid Systems)

```

Set  $X/\sim = \bigcup_q \mathcal{S}_q$ 
for  $q \in X_D$ 
  while  $\exists P, P' \in \mathcal{S}_q$  such that  $\emptyset \neq P \cap \text{Pre}_\tau(P') \neq P$ 
    Set  $P_1 = P \cap \text{Pre}_\tau(P')$ ;  $P_2 = P \setminus \text{Pre}_\tau(P')$ 
    refine  $\mathcal{S}_q = (\mathcal{S}_q \setminus \{P\}) \cup \{P_1, P_2\}$ 
  end while
end for

```

A few comments are in order here. The key problem is to investigate how the flow of $F(q, \cdot)$ interacts with the sets $\mathcal{S}_q$ for a single discrete state q . This requires that the trajectories of the vector field $F(q, \cdot)$ have “nice” intersection properties with such sets. Since the goal is to obtain finite partitions, it will become important that we restrict the study to classes of sets with good “finiteness” properties, for example, sets with finitely many connected components. In the subsequent sections we identify classes of sets and vector fields which exhibit such properties and for which **Algorithm 2** terminates.

One can also view the partitions in the algorithm as a way of discretizing the system trajectories. This suggests studying the continuous transitions by looking only at the points at which the trajectories move from one set in $\mathcal{S}_q$ to an “adjacent” one. This is in general not possible because sets could have rather pathological boundaries (see also Example 4). We will see in the next section that subanalytic sets are free from such pathologies and that in fact one can formalize the idea of trajectory discretization associated to the partition in that case.

We conclude this section with an example that shows that, even in apparently simple situations, **Algorithm 2** does not terminate.

Example 1. Let F be the linear vector field $\begin{pmatrix} -1 & 1 \\ -1 & -1 \end{pmatrix} \mathbf{x}$ on $\mathbb{R}^2$. Assume the partition of $\mathbb{R}^2$ consists of the following three sets (see Figure 1): $P_1 = \{(x, 0) : 0 \leq x \leq 4\}$, $P_2 = \{(x, 0) : -4 \leq x < 0\}$, $P_3 = \mathbb{R}^2 \setminus (P_1 \cup P_2)$. The integral curves of F are spirals moving away from the origin. The first iteration of the algorithm partitions P_2 into $P_4 = P_2 \cap \text{Pre}_\tau(P_1) = \{(x, 0) : x_1 \leq x < 0\}$ and

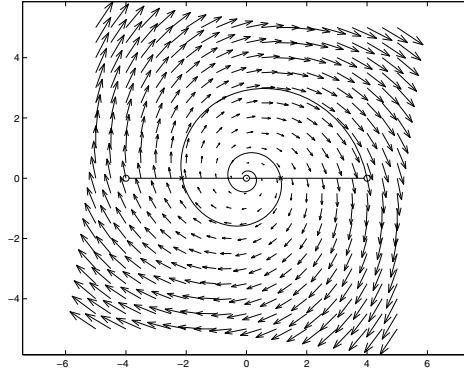


Fig. 1. Algorithm 2 does not terminate

$P_2 \setminus \text{Pre}_\tau(P_1)$. Here $x_1 < 0$ is the x -coordinate of the first intersection point of the spiral through $(4, 0)$ with P_2 . The second iteration subdivides P_1 into $P_5 = P_1 \cap \text{Pre}_\tau(P_4) = \{(x, 0) : 0 \leq x \leq x_2\}$ and $P_1 \setminus \text{Pre}_\tau(P_4)$ where $x_2 > 0$ is the x -coordinate of the next point of intersection of the spiral with P_1 . This process continues indefinitely since the spiral intersects P_1 in infinitely many points, and therefore the algorithm does not terminate.

4 Subanalytic Sets and Stratifications

In this section we describe some fundamental properties of *subanalytic sets* (see [6,13,21] for more details). A differentiable manifold is *real analytic* (C^ω) if the transition maps between local coordinate charts are analytic functions on their domains (which are open subsets of $\mathbb{R}^n$). An *embedded submanifold* S of a manifold M is a topological subspace of M together with a differentiable structure such that the inclusion from S into M is a smooth immersion (i.e. has full rank at every point). A vector field F on the real analytic manifold M is *analytic* if its coordinates in any local chart are analytic. If F is an analytic vector field then any integral curve of F is analytic.

Let M and N be real analytic manifolds and let $C^\omega(M, N)$ denote the set of analytic functions from M into N . If $f \in C^\omega(M, N)$ we say f is of class C^ω . Given an analytic manifold U , we denote by $\Sigma(C^\omega(U, \mathbb{R}))$ the Boolean algebra generated by the sets of the form $\{x : f(x) = 0\}$ or $\{x : f(x) > 0\}$, where $f \in C^\omega(U, \mathbb{R})$.

Definition 2. Let M be a real analytic manifold. A subset A of M is *semi-analytic* in M if for every $p \in M$, there is an open neighborhood U of p in M such that $U \cap A \in \Sigma(C^\omega(U, \mathbb{R}))$. If $A \subseteq M$ is *semianalytic* in M we write $A \in \text{SMAN}(M)$.

Definition 3. Let M be a real analytic manifold. Define $\text{SBAN}_{rc}(M)$ and $\text{SBAN}(M)$ by

1. $A \in \text{SBAN}_{rc}(M)$ if and only if there is (N, f, A^*) such that N is a real analytic manifold, $f \in C^\omega(N, M)$, $A^* \in \text{SMAN}(N)$, A^* is relatively compact and $A = f(A^*)$;
2. $A \in \text{SBAN}(M)$ if and only if A is the union of a locally finite collection of members of $\text{SBAN}_{rc}(M)$. (A collection of sets $\mathcal{C}$ is locally finite if any compact set intersect only finitely many sets in $\mathcal{C}$.)

We say that A is *subanalytic in M* if $A \in \text{SBAN}(M)$. It is easy to see that $A \in \text{SBAN}_{rc}(M)$ if and only if A is subanalytic in M and relatively compact. The following properties of subanalytic sets are easily derived from the definitions.

1. $\text{SBAN}(M)$ is closed under locally finite unions and intersections.
2. If $A \in \text{SBAN}(M)$ and $f: M \rightarrow N$ is of class C^ω and proper on $\overline{A}$, the closure of A , then $f(A) \in \text{SBAN}(N)$. (A function f is *proper* if $f^{-1}(K)$ is compact whenever K is.)
3. If $A \in \text{SBAN}(N)$ and $f: M \rightarrow N$ is of class C^ω , then $f^{-1}(A) \in \text{SBAN}(M)$.

The following two properties require more subtle proofs, but they give the first indication that this will be a suitable class of sets for our studies.

4. If $A \in \text{SBAN}(M)$ then $M \setminus A \in \text{SBAN}(M)$.
5. A subanalytic set has a locally finite number of connected components, each of which is subanalytic.

Example 2. Points are subanalytic, and so is any locally finite union of points, for example $\mathbb{Z}^n$ as subset of $\mathbb{R}^n$. The empty set and M are both in $\text{SBAN}(M)$. Let $a, b \in \mathbb{R}$, $a < b$, then $[a, b]$, $[a, b)$, $(a, b]$ and (a, b) are subanalytic in $\mathbb{R}$. The open ball $B(p, r)$ centered at p of radius r in $\mathbb{R}^n$ is in $\text{SBAN}(\mathbb{R}^n)$.

Definition 4. Let M be a real analytic manifold. An analytic (C^ω) stratification of M is a partition $\mathcal{S}$ of M with the following properties:

1. each $S \in \mathcal{S}$ is a connected, real analytic, embedded submanifold of M ,
2. $\mathcal{S}$ is locally finite,
3. given two sets $S, P \in \mathcal{S}$, $P \neq S$, such that $S \cap \overline{P} \neq \emptyset$ then $S \subset \overline{P}$ and $\dim S < \dim P$.

The sets in a stratification are called strata.

The central result on stratifications for our analysis is the following. For a proof see [20].

Theorem 1. Let $\mathcal{A}$ be a locally finite family of nonempty subanalytic subsets of a real analytic manifold M . For each $A \in \mathcal{A}$, let $F(A)$ be a finite set of real analytic vector fields on M . Then there exists a subanalytic stratification $\mathcal{S}$ of M , compatible with $\mathcal{A}$, and having the property that, whenever $S \in \mathcal{S}$, $S \subseteq A$, $A \in \mathcal{A}$, $X \in F(A)$, then either (i) F is everywhere tangent to S or (ii) F is nowhere tangent to S . ($\mathcal{S}$ is compatible with $\mathcal{A}$ is every set in $\mathcal{A}$ is a union of sets in $\mathcal{S}$.)

Theorem 1 is illustrated by the following example.

Example 3. Let F be the following analytic vector field on $\mathbb{R}^2$

$$\begin{aligned}\dot{x} &= x^2 + y^2 \\ \dot{y} &= 0\end{aligned}$$

which has an isolated equilibrium at the origin and points in the positive x -direction otherwise. Consider the following two subanalytic sets

$$\begin{aligned}S_1 &= \{(x, y) \in \mathbb{R}^2 \mid y \geq 0 \text{ and } (x-1)^2 + y^2 = 1\} \\ S_2 &= \{(x, y) \in \mathbb{R}^2 \mid y = 0 \text{ and } 0 \leq x \leq 2\}\end{aligned}$$

shown in Figure 2. A subanalytic stratification of $\mathbb{R}^2$ which is compatible with the sets S_1, S_2 and the vector field F is also shown in Figure 2. It consists of

- 0-dimensional strata
 - $P_1 = (0, 0)$, $P_2 = (2, 0)$, and $P_3 = (1, 1)$
- 1-dimensional strata
 - $C_1 = \{(x, y) \in \mathbb{R}^2 \mid y = 0 \text{ and } 0 < x < 2\}$
 - $C_2 = \{(x, y) \in \mathbb{R}^2 \mid y > 0 \text{ and } 1 < x < 2 \text{ and } (x-1)^2 + y^2 = 1\}$
 - $C_3 = \{(x, y) \in \mathbb{R}^2 \mid y > 0 \text{ and } 0 < x < 1 \text{ and } (x-1)^2 + y^2 = 1\}$
- 2-dimensional strata
 - $D_1 = \{(x, y) \in \mathbb{R}^2 \mid y > 0 \text{ and } (x-1)^2 + y^2 < 1\}$
 - $D_2 = \mathbb{R}^2 \setminus \{P_1, P_2, P_3, C_1, C_2, C_3, D_1\}$

Notice that the vector field is tangent to P_1 since it is an equilibrium as well as to C_1, D_1 and D_2 . The vector field is transverse to all the other strata. Moreover, $S_1 = P_1 \cup P_2 \cup P_3 \cup C_2 \cup C_3$ and $S_2 = P_1 \cup P_2 \cup C_1$.

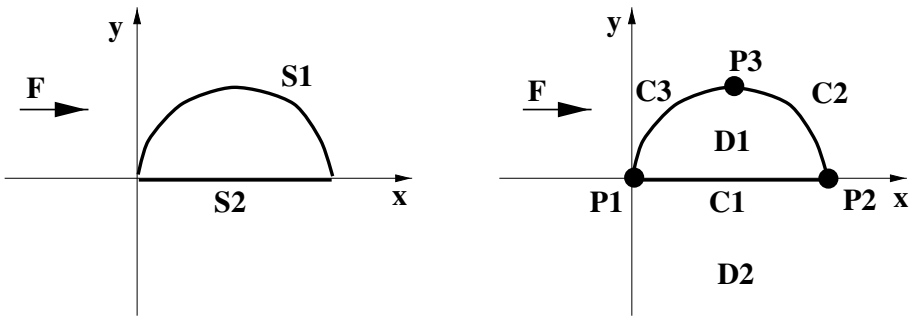


Fig. 2. Subanalytic stratification example

In view of the above properties we will restrict our study to hybrid systems for which the relevant sets are all relatively compact and subanalytic.

Assumption 1 : For each discrete state q the collection $\mathcal{A}_q$ consists of relatively compact subanalytic sets. In particular, we assume there exists a compact set K such that if $A \in \mathcal{A}_q$ then $A \subseteq K$.

The partition $\mathcal{S}_q$ which serves as the initialization step of **Algorithm 2** can now be assumed to be a subanalytic stratification compatible with $\mathcal{A}_q$ and the vector field $F(q, \cdot)$ (as given by Theorem 1).

The following proposition illustrates some of the good intersection properties that analytic curves have with subanalytic sets. The “finiteness” property indicated in the proposition makes it possible to define transitions between adjacent strata in a natural way.

Proposition 1. *Let I be an open interval, M a real analytic manifold and $\gamma: I \rightarrow M$ a real analytic function. Let $\mathcal{S}$ be a C^ω stratification of M by subanalytic sets. If $[a, b] \subset I$ then there exists a finite partition $\{x_1, \dots, x_n\}$ of $[a, b]$ with the property that for each $i = 1, \dots, n-1$ there exists a stratum $S_i \in \mathcal{S}$ such that $\gamma((x_i, x_{i+1})) \subseteq S_i$.*

Proof. The family $\mathcal{I} = \{\gamma^{-1}(S) \cap [a, b] : S \in \mathcal{S}\}$ is a finite partition of $[a, b]$ by subanalytic sets. Each such set consists of a finite number of points and open intervals. Using all such points and the endpoints of such intervals gives the desired partition.

The following example shows the type of pathological situations that can be encountered if the assumption on subanalyticity is even slightly relaxed.

Example 4. Consider the stratification of $\mathbb{R}^2$ by the following five sets:

$$\begin{aligned} S_1 &= \{(0, 0)\} \\ S_2 &= \left\{ (x, y) : x > 0 \wedge y = x \sin \frac{1}{x} \right\} \\ S_3 &= \left\{ (x, y) : x < 0 \wedge y = x \sin \frac{1}{x} \right\} \\ S_4 &= \left\{ (x, y) : x \neq 0 \wedge y > x \sin \frac{1}{x} \right\} \bigcup \{(0, y) : y > 0\} \\ S_5 &= \left\{ (x, y) : x \neq 0 \wedge y < x \sin \frac{1}{x} \right\} \bigcup \{(0, y) : y < 0\} \end{aligned}$$

Notice that S_1 , S_2 and S_3 form the graph of the function $f(x) = x \sin \frac{1}{x}$ ($f(0) = 0$), while S_4 and S_5 denote the region above and the below the graph, respectively. Each set is a C^ω , embedded submanifold of $\mathbb{R}^2$ and they clearly satisfy the condition on the dimension of the strata in the closure of other strata. Finally, consider the constant vector field $F = \frac{\partial}{\partial x}$. Then the integral curve γ of F through $(0, 0)$ is the x -axis (parameterized by x itself). Therefore, the image by γ of any interval containing 0 intersects both S_4 and S_5 an infinite number of times. This is reminiscent of the undesirable zeno property which allows an infinite number of switches in finite time.

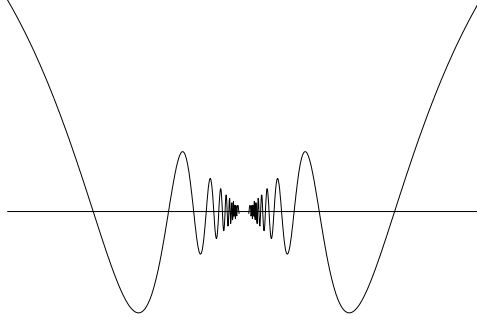


Fig. 3. Infinite crossings on a compact interval

Since the algorithm considers one discrete state at a time, we will simplify the notation by assuming that the discrete state q is fixed and drop it as a subscript. In particular we will consider a vector field F and a stratification $\mathcal{S}$ of X_C by subanalytic sets as provided by Theorem 1. By $X_C/\sim$ we will mean the partition of X_C induced by $\mathcal{S}$. We will denote by γ_x the integral curve of F which passes through x at time 0, i.e. with $\gamma_x(0) = x$.

We now proceed to formalize the notion of a discretization of the continuous transitions relative to a given partition $\mathcal{S}$. We do this mainly it simplifies the arguments in the proof of the main theorem (Theorem 2). In addition it supports the intuitive picture we have that a trajectory can be decomposed as a concatenation of pieces in each of the sets in $\mathcal{S}$.

Definition 5 (Transition relative to $\mathcal{S}$: version 1). *Given $x, y \in X_C$ we say $x \xrightarrow{\mathcal{S}} y$ iff there is $t > 0$ such that $\gamma_x(t) = y$ and there exists $S \in \mathcal{S}$ such that $\gamma_x(s) \in S$ for $0 < s < t$ and at least one of x, y is in S .*

To clarify this concept and to facilitate further discussions and proofs we introduce additional definitions.

Definition 6. *Given two subsets S_1, S_2 of X_C , and a real analytic curve $\gamma : I \rightarrow X_C$ where I is an open interval, we say that γ leaves S_1 through S_2 (or enters S_2 from S_1) if one of the following exiting conditions is satisfied:*

- E1** *there exist $a, b \in I$, $a < b$, such that $\gamma(t) \in S_1$ for all $t \in (a, b)$ and $\gamma(b) \in S_2$*
- E2** *there exist $a, b \in I$, $a < b$, such that $\gamma(a) \in S_1$ and $\gamma(t) \in S_2$ for all $t \in (a, b)$.*

*When $x \in S_1$ we say that γ_x leaves S_1 through S_2 if either **E1** or **E2** holds with $a = 0$.*

The following proposition is a simple application of Proposition 1 and shows that Definition 6 covers all possible “exiting” situations for strata of $\mathcal{S}$.

Proposition 2. *Let $S_1 \in \mathcal{S}$ and γ be as above. If there exists $t_0, t_1 \in I$ such that $\gamma(t_0) \in S_1$ and $\gamma(t_1) \notin S_1$ then there exists a stratum $S_2 (\neq S_1)$ such that either **E1** or **E2** holds.*

It is clear from Definition 6 that in case **E1**, $S_2 \cap \overline{S_1} \neq \emptyset$. By the properties of stratifications, we conclude $S_2 \subset \overline{S_1}$ and $\dim S_2 < \dim S_1$. Therefore, the flow exits the stratum S_1 through a stratum of lower dimension. Similarly in case **E2**, $S_1 \subset \overline{S_2}$ and $\dim S_1 < \dim S_2$ and the flow enters S_2 from a stratum of lower dimension. The following proposition further clarifies the possible exit situations.

Definition 7. *We call a stratum $S \in \mathcal{S}$ tangential if the vector field F is tangent to S at every point of S . We call a stratum transversal otherwise.*

Proposition 3. *Let S_1, S_2 be strata in $\mathcal{S}$ and γ an integral curve of F which leaves S_1 through S_2 . Then one (and only one) of the following holds:*

1. *condition **E1** holds, S_1 is a tangential stratum and S_2 is a transversal stratum.*
2. *condition **E2** holds, S_1 is a transversal stratum and S_2 is a tangential stratum.*

We can now give the alternative definition of relative transitions.

Definition 8 (Transition relative to $\mathcal{S}$: version 2). *For each $x \in X_C$ let $S(x)$ denote the unique stratum in $\mathcal{S}$ which contains x . Given $x, y \in X_C$ we say $x \xrightarrow{\mathcal{S}} y$ iff γ_x leaves $S(x)$ through $S(y)$.*

It is clear from Proposition 1 that $x \xrightarrow{\tau} y$ iff there exist $x_1, \dots, x_n$ such that $x \xrightarrow{\mathcal{S}} x_1 \xrightarrow{\mathcal{S}} \dots \xrightarrow{\mathcal{S}} x_n \xrightarrow{\mathcal{S}} y$. We will denote the *Pre* operator associated to $\xrightarrow{\mathcal{S}}$ by $Pre_{\mathcal{S}}$. The above remark also implies that we can substitute $Pre_{\mathcal{S}}$ for Pre_{τ} in **Algorithm 2** in the sense that if the algorithm terminates using $Pre_{\mathcal{S}}$ then it also terminates when using Pre_{τ} .

As the stratification Theorem 1 shows, issues of transversality of trajectories can be analyzed within the context of subanalytic sets and analytic vector fields. However, the study of continuous transitions requires that we investigate the global behavior of trajectories. In general, trajectories of analytic vector fields (and much less their full flows) are not subanalytic. Identifying vector fields whose flows belong to a suitable class is the main obstacle in the study of bisimulations of hybrid systems. Recent developments in logic model theory provide some answers as well as suggest the proper context in which to carry on further studies.

5 Model Theory

Model theory studies structures through properties of their definable sets (see [14,23] for general background). The basic structures of interest for this

paper are that of the real numbers as a complete ordered field, symbolized by $(\mathbb{R}, +, -, \times, <, 0, 1)$, and its extensions. Every such structure L has an associated language $\mathcal{L}$ of formulas. The (first order) formulas over $\mathcal{L}$ are the well-formed logical expressions obtained by using logical connectives, quantifiers $\exists$ $\forall$, real numbers as constants, the operations of additions and multiplication, and the relations $<$ and $=$ (quantification is allowed over variables). All formulas will be interpreted over the real numbers. A *definable set* in the language $\mathcal{L}$ (or of the structure L) is a subset of $\mathbb{R}^n$ (for some n) of the form $\{(a_1, \dots, a_n) \in \mathbb{R}^n : \Phi(a_1, \dots, a_n)\}$, where $\Phi(x_1, \dots, x_n)$ is a formula in $\mathcal{L}$ and $x_1, \dots, x_n$ are free (i.e. not quantified) variables in Φ . A function f is definable if its graph is a definable set.

While many of the concepts here apply to more general structures, in all that follows we consider only structures over the real numbers.

Definition 9. *The theory of $\mathcal{L}$ is o-minimal (“order minimal”) if every definable subset of $\mathbb{R}$ is a finite union of points and intervals (possibly unbounded).*

Tarski [22] was interested in the extension of the theory of the real numbers by the exponential function, $(\mathbb{R}, +, -, \times, <, 0, 1, \exp)$ (i.e., there is an additional symbol in the language for the exponential function). We denote this structure by $\mathbb{R}_{\exp}$. While such theory does not admit elimination of quantifiers, it was shown in [25] that such theory is model complete, which in turns implies that it is o-minimal. Another important extension is obtained as follows. Assume f is a real-analytic function in a neighborhood of the cube $[-1, 1]^n \subset \mathbb{R}^n$. Let $\hat{f}: \mathbb{R}^n \rightarrow \mathbb{R}$ be the function defined by

$$\hat{f}(x) = \begin{cases} f(x) & \text{if } x \in [-1, 1]^n \\ 0 & \text{otherwise} \end{cases}$$

We call such functions *restricted analytic functions*. The structure $\mathbb{R}_{\exp, \text{an}} = (\mathbb{R}, +, -, \times, <, 0, 1, \exp, \{\hat{f}\})$ is then an extension of $\mathbb{R}_{\exp}$ where there is a symbol for each restricted analytic function. One reason this structure is relevant for this paper is that all relatively compact subanalytic sets are definable in $\mathbb{R}_{\exp, \text{an}}$. Moreover, if F is a linear vector field in $\mathbb{R}^n$ with real eigenvalues, then the trajectories of F are definable in $\mathbb{R}_{\exp, \text{an}}$. In [24], it was shown that $\mathbb{R}_{\exp, \text{an}}$ is also o-minimal. Finally, there are a few consequences of o-minimality that are crucial for our results. We list them below under one proposition. The proofs are contained in the various references mentioned above.

Proposition 4. *Assume L is an o-minimal structure. Then*

1. *Any definable set has a finite number of connected components, each of which is a definable set.*
2. *If A is definable, then so is its (topological) closure. Moreover, $\dim \text{Fr}(A) < \dim A$, where $\text{Fr}(A) = \overline{A} \setminus A$ is the frontier of A and the dimension of a set $B \subset \mathbb{R}^n$ is the maximum integer d for which there is an embedded C^1 manifold of $\mathbb{R}^n$ contained in B .*

3. Given definable sets $A_1, \dots, A_k$ in $\mathbb{R}^n$ (and for any integer p), there is a finite C^p stratification of $\mathbb{R}^n$ compatible with $\{A_1, \dots, A_k\}$. In fact, for the structure $\mathbb{R}_{\text{exp,an}}$ the strata are definable (real) analytic manifolds.

We are now ready to apply these results to prove that **Algorithm 2** terminates for certain classes of planar systems.

6 Finiteness Results

In this section we use the model theoretic tools of Section 5 in order to obtain classes of systems for which the Bisimulation Algorithm of Section 3 terminates.

Recall that given the family of sets $\mathcal{A}$ as in Assumption 1, and the vector field F we first obtain a stratification $\mathcal{S}$ compatible with $\mathcal{A}$ as given by Theorem 1. We will also assume that $\mathcal{S}$ is compatible with a compact subanalytic set K which contains all sets in $\mathcal{A}$. We define $\mathcal{S}_K = \{S \in \mathcal{S} : S \cap K \neq \emptyset\}$ (which is therefore finite).

Theorem 2. *Let $X_C = \mathbb{R}^2$, F be the linear vector field Ax and assume that the eigenvalues of A are real. Then the bisimulation algorithm for hybrid systems (**Algorithm 2**), initialized with $\mathcal{S}_K$, terminates.*

Proof. We will consider the case when the origin is the only equilibrium of F . (The other cases require minor modifications.) We assume without loss of generality that $\{(0, 0)\} \in \mathcal{S}_K$.

As indicated in Section 3 it suffices to study only the evolution of the continuous variables and use $Pre_{\mathcal{S}}$ in **Algorithm 2**. To simplify notation we will simply refer to it as Pre . In order to show that the bisimulation algorithm terminates we will construct a finite refinement of $\mathcal{S}_K$ which is “invariant” under the Pre operation and which is a refinement of $X_C/\sim$ at each step.

For each stratum $S \in \mathcal{S}_K$ with $(0, 0) \in \overline{S}$ we consider the set

$$S_{\infty} = \{x \in S : \forall t \geq 0 \ \gamma_x(t) \in S\}$$

As mentioned earlier, since the eigenvalues of A are real, the flow of F , $\Phi(x, t) = \gamma_x(t) = e^{tA}x$ is definable in $\mathbb{R}_{\text{exp,an}}$ (the entries in e^{tA} involve polynomials and real exponential functions). Therefore, the set S_{∞} is definable. For each stratum T of dimension one with $T \subset \overline{S}$, $T \neq S$, we consider the set

$$T_* = \{x \in T : \gamma_x \text{ leaves } T \text{ through } S_{\infty}\}$$

The set T_* is also definable in $\mathbb{R}_{\text{exp,an}}$ and therefore can be written as a finite, disjoint union of definable sets each of which is either a point or homeomorphic to an open interval. We may assume, by refining the original $\mathcal{S}_K$ if necessary that the finitely many points in the decomposition of T_* are already strata of $\mathcal{S}_K$.

For each $x \in \mathbb{R}^2$ let Γ_x denote the trajectory of F passing through x , that is

$$\Gamma_x = \{\gamma_x(t) : t \in \mathbb{R}\}.$$

For each stratum $S \in \mathcal{S}$ and $x \in S$, let $\Gamma_x(S)$ denote the connected component of $\Gamma_x \cap S$ which contains x . It is clear, from the definition of S_∞ , that if $x \in S_\infty$ then $\Gamma_x(S) \subset S_\infty$. From this it follows that if $x \in T$ and γ_x leaves T through S then γ_x either leaves T through S_∞ or leaves T through $S \setminus S_\infty$.

Let $\{p_1\}, \dots, \{p_l\}$ be all the 0-dimensional strata of $\mathcal{S}_K$. Notice that for each i, j , if $\Gamma_{p_i} \cap \Gamma_{p_j} \neq \emptyset$, then $\Gamma_{p_i} = \Gamma_{p_j}$. We will eliminate redundancies and assume that the Γ_{p_i} are pairwise disjoint. For each set $S \in \mathcal{S}_K$ and each Γ_{p_i} , the sets $S \cap \Gamma_{p_i}$ and $S \setminus \cup_i \Gamma_{p_i}$ are definable in $\mathbb{R}_{\text{exp,an}}$ (Intuitively, these sets are partitions of S “in the direction of the flow of F ”). By o-minimality, we get that each such set has a finite number of connected components. Let $\mathcal{B}$ denote the (finite) collection of all such connected components. The collection $\mathcal{B}$ is then a partition of K compatible with $\mathcal{S}$ (every set in $\mathcal{S}$ is a union of sets in $\mathcal{B}$).

Claim: At each step of the bisimulation algorithm, $\mathcal{B}$ is compatible with $M/\sim$.

The claim shows that $\mathcal{B}$ is finer than all partitions obtained at each step. Since $\mathcal{B}$ is finite, this proves that the algorithm terminates.

To prove the claim we first show that if $B_i \in \mathcal{B}$ for $i = 1, \dots, n$ then

$$Pre(\cup B_i) = \cup Pre(B_i) \quad (4)$$

We will call a set $B \in \mathcal{B}$ tangential if B is contained in a tangential stratum of $\mathcal{S}$ (i.e. B is a connected component of either $S \cap \Gamma_q$ or $S \setminus \cup \Gamma_{p_i}$ with S tangential). The set B will be called transversal otherwise. Notice that if B is tangential and $x \in B$ then $\Gamma_x(S(x)) \subset B$.

Let $x \in Pre(B_i)$ for some $i = 1, \dots, n$ and $x \notin B_i$. Suppose $\gamma_x(t) \in S(x)$ for $0 \leq t < \delta$ and $\gamma_x(\delta) \in B_i$ (i.e. exit condition **E1**). In particular, $S(x)$ is a tangential stratum. If $\gamma_x(t) \notin \cup B_i$ for $t < \delta$, then $x \in Pre(\cup B_i)$. If $\gamma_x(t) \in \cup B_i$ for some $t < \delta$, then for some j , B_j is tangential, so $\Gamma_x(S(x)) \subset B_j$ and $x \in Pre(\cup B_i)$. If, instead, $\gamma_x(t) \in B_i$ for $0 < t < \delta$ (exit condition **E2**), then clearly $x \in Pre(\cup B_i)$.

Conversely, let $x \in Pre(\cup B_i)$. If $\gamma_x(t) \in S(x)$ for $0 \leq t < \delta$, $\gamma_x(\delta) \in \cup B_i$, let i_0 be such that $\gamma_x(\delta) \in B_{i_0}$. Then $x \in Pre(B_{i_0}) \subset \cup Pre(B_i)$. If, instead, $\gamma_x(t) \in \cup B_i$ for $0 < t < \delta$, then there is a $\delta_0 > 0$ and a B_{i_0} which contains $\gamma_x(t)$ for $0 < t < \delta_0$ (here we used o-minimality again to conclude that Γ_x intersects each B_i in a finite disjoint union of points and arcs). Therefore, $x \in Pre(B_{i_0})$. This concludes the proof of (4).

By construction, $\mathcal{B}$ is compatible with $\mathcal{S}_K$. At each step of the bisimulation algorithm we need to show that if $B = \cup_{i=1}^n B_i$ and $B' = \cup_{j=1}^m B'_j$ with $B_i, B'_j \in \mathcal{B}$ then $B \cap Pre(B')$ is again a finite union of sets in $\mathcal{B}$. Based on (4) it will suffice to show that for $B, B' \in \mathcal{B}$, either $B \cap Pre(B') = \emptyset$ or $B \cap Pre(B') = B$.

We consider several cases. The set B is of one of the two forms: (a) a connected component of $S \cap \Gamma_{p_i}$, or (b) a connected component of $S \setminus \cup \Gamma_{p_i}$.

If S is 0-dimensional there is nothing to show because B contains a single point.

If S is 1-dimensional and B is of type (a), then either S is transversal and B consists of a single point or S is tangential and so $B = \Gamma_x(S)$ for any $x \in B$.

The first case is again clear. In the second case, if there is $x \in B \cap \text{Pre}(B')$ then there exists $\delta > 0$ such that $\gamma_x(t) \in S$ for $0 \leq t < \delta$ and $\gamma_x(\delta) \in B'$. But then for all $y \in \Gamma_x(S)$, γ_y leaves S through B' . So $B = \Gamma_x(S) \subset \text{Pre}(B')$.

If S is 1-dimensional and B is of type (b) then we again consider separately the cases when S is tangential and when S is transversal. In the first case we proceed as before. Assume now, that S is transversal. Notice that if $x \in B \cap \text{Pre}(B')$ then Γ_x intersects both B and B' . Therefore B' is also a connected component of $S' \setminus \cup \Gamma_{p_i}$ (for some S'). By transversality, γ_x leaves S into S' under exit condition **E2** and so $S \subset \text{Fron}(S') (= \overline{S'} \setminus S')$ and S' is 2-dimensional. By continuity of the flow of F , there is an open neighborhood N of x such that for $y \in N \cap B$, γ_y leaves S through S' . Moreover, since there are finitely many Γ_{p_i} we may assume (by taking N smaller) that γ_y leaves S through B' . We have then showed that the set $E = \{x \in B : \gamma_x \text{ leaves } S \text{ through } B'\}$ is open in B . Suppose $E \neq B$. Then there is $y \in B$ in the frontier of E . We can find a neighborhood W of y such that $W \cap \Gamma_{p_i} = \emptyset$ for all i . Since S' is open in $\mathbb{R}^2$, and S is transversal, we can find a neighborhood $W_0 \subset W$ of y and $\varepsilon > 0$ such that for $z \in W_0 \cap S$ and $0 < t < \varepsilon$ we have $\gamma_z(t) \in W \cap S'$. But then every such z belongs to E . This contradicts the fact that y is a frontier point. Therefore, E is also closed in B and so it must equal B (since B is connected). We conclude in this case that $B = B \cap \text{Pre}(B')$.

There is only one case remaining: S of dimension 2 (and hence tangential). If B is of type (a) then $\Gamma_x(S) = B$ and we are done as before.

Assume then that B is a connected component of $S \setminus \cup \Gamma_{p_i}$, B' a connected component of $S' \setminus \cup \Gamma_{p_i}$, S' is transversal, and $\dim S' = 1$. (The case with S' 0-dimensional is excluded since in that case $S' \cap \Gamma_{p_i} \neq \emptyset$ for some i .)

Let $x \in B \cap \text{Pre}(B')$ and assume there is $y \in B \setminus \text{Pre}(B')$. We want to show that this leads to a contradiction. Let $\alpha : [0, 1] \rightarrow B$ be a curve connecting x to y . Let t_0 be the smallest $t \in [0, 1]$ such that $\gamma_{\alpha(t)}(s) \notin B'$ for some $s > 0$. If $\gamma_{\alpha(t_0)}(s) \in S$ for all $s > 0$ then $\alpha(t_0) \in S_\infty$. By the choice of t_0 we in fact have $\alpha(t_0) \in \Gamma_{p_0}$ for some p_0 (see the initial subdivision caused by S_∞). But this contradicts the fact that B is of type (b). Assume then that $\gamma_{\alpha(t_0)}(s) \notin S$ for some $s > 0$. For each $t \in [0, t_0]$ let $s(t)$ be the smallest s such that $\gamma_{\alpha(t)}(s) \notin S$. For each $t \in [0, t_0]$ set $p(t) = \gamma_{\alpha(t)}(s(t))$. There are two possibilities: either $p(t_0) \in S'$ or $p(t_0) \in \overline{S'} \setminus S$.

In the first case choose a local chart (N, φ) centered at $p(t_0)$ so that in φ -coordinates we have $N \cap S' = N \cap B' = \{(x, 0)\}$ and $N \cap S = \{(x, y) : y > 0\}$ (therefore F points into the lower half plane at every point of $N \cap B'$). By continuity of the flow and transversality, we still have that $\gamma_{\alpha(t)}$ crosses $N \cap B'$ from the upper to the lower half plane for $t_0 < t < t_0 + \epsilon$. But this contradicts the choice of t_0 .

In the second case, we have $p(t_0) \in \Gamma_{q_0}$ for some q_0 . But this contradicts the fact that B is of type (b).

All this implies that every y in B must also be in $\text{Pre}(B')$. That is, $B = B \cap \text{Pre}(B')$. This concludes the proofs of the claim and the theorem.

As the proof above suggests the termination of the algorithm depends on the fact that the integral curves of the vector field intersects relatively compact sub-analytic sets in at most finitely many points. This allows us to get the following generalization.

Theorem 3. *If F is an analytic vector field in $\mathbb{R}^2$ which admits an analytic family of first integrals, then the bisimulation algorithm terminates. (Here, by an analytic family of first integrals we mean a non-constant (real) analytic function $f: \mathbb{R}^2 \rightarrow \mathbb{R}$ such that for each trajectory γ of F the function $f(\gamma(t))$ is constant.)*

Proof. Notice that each level curve of f is an analytic set and therefore its intersection with any relatively compact definable set (in $\mathbb{R}_{\text{exp,an}}$) is definable in $\mathbb{R}_{\text{exp,an}}$. The proof then follows the lines of the previous one but replacing the sets Γ_{p_i} , with the corresponding level set of f (level sets of f are at most 1-dimensional since f is not constant on any open set).

Corollary 1. *If F is a linear vector field in $\mathbb{R}^2$ with purely imaginary eigenvalues and $\mathcal{S}_K$ is as in the theorem, then the bisimulation algorithm terminates.*

Proof. Unless $A = 0$, in which case the result is trivial, there exists an (invertible) matrix P such that $\|Px\|^2$ is constant along trajectories of F .

Corollary 2. *If F is an analytic Hamiltonian vector field in $\mathbb{R}^2$ and $\mathcal{S}_K$ is as above, then the bisimulation algorithm terminates.*

Proof. The Hamiltonian is constant along the trajectories.

Remark 1. As is clear from the proofs above, the key is that all the objects involved (the vector field F , the initial family of sets, the flow of F) be definable in some o-minimal extension of the field of real numbers. We presented above just two specific instances of such a situation which can be easily characterized. A more recent o-minimal extension of the reals, by so called Pfaffian functions, was found in [26].

7 Conclusions

In this paper, we presented new classes of planar hybrid systems with finite bisimulations. This was achieved by combining the geometric framework of subanalytic sets with model theoretic concepts from mathematical logic. The mathematical tools used in this paper provide the natural platform for studying decidability of computational algorithms for hybrid systems.

Issues for future research, include the extension of these results to $\mathbb{R}^n$ as well as to hybrid systems whose relevant sets and flows are definable in o-minimal structures. In addition, the issue of decidability requires not only termination of the bisimulation algorithm but also constructive decision methods for each step.

Even though decision methods exist for $(\mathbb{R}, +, -, \times, <, 0, 1)$ [22], it is not known if the theory of $\mathbb{R}_{\text{exp}}$ is decidable, although in [16] it was shown that it would be a consequence of Schanuel's conjecture in number theory. The results we obtained in this paper suggest how to find some restricted classes of vector fields for which the algorithm is constructive. Indeed, if all the relevant sets are semialgebraic (for example if F is a Hamiltonian vector field on the plane with a polynomial Hamiltonian and the initial conditions, guards, etc., are semialgebraic), then they are definable in $(\mathbb{R}, +, -, \times, <, 0, 1)$. Such a decidability result is obtained in [9].

References

1. R. Alur, C. Coucoubetis, N. Halbwachs, T.A. Henzinger, P.H. Ho, X. Nicolin, A. Olivero, J. Sifakis, and S. Yovine, *The algorithmic analysis of hybrid systems*, Theoretical Computer Science **138** (1995), 3–34.
2. R. Alur and D.L. Dill, *A theory of timed automata*, Theoretical Computer Science **126** (1994), 183–235.
3. R. Alur, T.A. Henzinger, and E.D. Sontag (eds.), *Hybrid systems III*, Lecture Notes in Computer Science, vol. 1066, Springer-Verlag, 1996.
4. P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry (eds.), *Hybrid systems II*, Lecture Notes in Computer Science, vol. 999, Springer-Verlag, 1995.
5. P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry (eds.), *Hybrid systems IV*, Lecture Notes in Computer Science, vol. 1273, Springer-Verlag, 1997.
6. E. Bierstone and P.D. Milman, *Semianalytic and subanalytic sets*, Inst. Hautes Études Sci. Publ. Math. **67** (1988), 5–42.
7. P.E. Caines and Y.J. Wei, *The hierarchical lattices of a finite state machine*, Systems and Control Letters **25** (1995), 257–263.
8. ———, *Hierarchical hybrid control systems: A lattice theoretic formulation*, IEEE Transactions on Automatic Control : Special Issue on Hybrid Systems **43** (1998), no. 4, 501–508.
9. K. Cerans and J. Viksna, *Deciding reachability for planar multi-polynomial systems*, Hybrid Systems III (R. Alur, T. Henzinger, and E.D. Sontag, eds.), Lecture Notes in Computer Science, vol. 1066, Springer Verlag, Berlin, Germany, 1996, pp. 389–400.
10. R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel (eds.), *Hybrid systems*, Lecture Notes in Computer Science, vol. 736, Springer-Verlag, 1993.
11. T.A. Henzinger, *Hybrid automata with finite bisimulations*, ICALP 95: Automata, Languages, and Programming (Z. Fülöp and F. Gécseg, eds.), Springer-Verlag, 1995, pp. 324–335.
12. T.A. Henzinger, P.W. Kopke, A. Puri, and P. Varaiya, *What's decidable about hybrid automata?*, Proceedings of the 27th Annual Symposium on Theory of Computing, ACM Press, 1995, pp. 373–382.
13. H. Hironaka, *Subanalytic sets*, In Number Theory, Algebraic Geometry, and Commutative Algebra, in honor of Y. Akizuki, Kinokuniya Publications, 1973, pp. 453–493.
14. W. Hodges, *A shorter model theory*, Cambridge University Press, 1997.

15. G. Lafferriere, G.J. Pappas, and S. Sastry, *Subanalytic stratifications and bisimulations*, Hybrid Systems : Computation and Control (T. Henzinger and S. Sastry, eds.), Lecture Notes in Computer Science, vol. 1386, Springer Verlag, Berlin, 1998, pp. 205–220.
16. A. Macintyre and A.J. Wilkie, *On the decidability of the real exponential field*, Kreiseliana: About and around Georg Kreisel, A.K. Peters, 1996, pp. 441–467.
17. O. Maler (ed.), *Hybrid and real-time systems*, Lecture Notes in Computer Science, vol. 1201, Springer-Verlag, 1997.
18. G.J. Pappas, G. Lafferriere, and S. Sastry, *Hierarchically consistent control systems*, Proceedings of the 37th IEEE Conference in Decision and Control, Tampa, FL, December 1998, Submitted.
19. A. Puri and P. Varaiya, *Decidability of hybrid systems with rectangular differential inclusions*, Computer Aided Verification, 1994, pp. 95–104.
20. Héctor J. Susmann, *Subanalytic sets and feedback control*, Journal of Differential Equations **31** (1979), no. 1, 31–52.
21. ———, *Real-analytic desingularization and subanalytic sets: An elementary approach*, Transactions of the American Mathematical Society **317** (1990), no. 2, 417–461.
22. A. Tarski, *A decision method for elementary algebra and geometry*, second ed., University of California Press, 1951.
23. D. van Dalen, *Logic and structure*, third ed., Springer-Verlag, 1994.
24. L. van den Dries and C. Miller, *On the real exponential field with restricted analytic functions*, Israel Journal of Mathematics **85** (1994), 19–56.
25. A. J. Wilkie, *Model completeness results for expansions of the ordered field of real numbers by restricted pfaffian functions and the exponential function*, Journal of the American Mathematical Society **9** (1996), no. 4, 1051–1094.
26. A.J. Wilkie, *A general theorem of the complement and some new o-minimal structures*, Preprint, 1997.

Extracting Stable Timed Automata from Switched Model Reference Systems

Michael Lemmon and Christopher J. Bett *

Dept. of Electrical Engineering
University of Notre Dame

Abstract. This paper discusses recent results on multiple linear agent control for systems satisfying a bounded amplitude performance constraint. The plant is assumed to be a linear parameter varying (LPV) system scheduled along a nominal parameter trajectory; in this respect, the control problem represents a plant operating between a prespecified set of operating conditions. Linear controllers are designed at setpoints along this scheduling trajectory to satisfy bounded amplitude performance constraints. This paper discusses an approach to analyze the switched system behavior under practical assumptions on the structure of the switching rule. The approach combines the scheduling parameter with LPV system properties to derive bounds on the switching behavior of the system. These estimates are then used to construct a logical model of the switched system behavior in the form of a timed automaton. In this respect, this paper presents a way of extracting logical models of continuous time system behavior.

1 Introduction

Complex systems are often built up through the aggregation of simpler subsystems. A manufacturing system, for example, may consist of several robots whose individual actions must be coordinated in a way that prevents deadlock and ensures the satisfactory completion of a specified task. Methods for the systematic design of such systems have only recently emerged. One of the major methods uses automatic verification methods applied to a timed transition system known as the timed automaton [1]. These verification methods determine whether or not the timed automaton satisfies a behavioural specification framed as a formula in the timed computational tree logic (TCTL) [2]. The verification method is based on real-time extensions of symbolic model checking methods [9].

An important first step in the analysis of such systems involves the extraction of timed automaton representations for the continuous-time system comprising the hybrid dynamical system. For example, in the robotic example outlined above, each robot can be characterized by a set of ordinary differential equations. Because the robot functions over a very large operating range, however, it

* We gratefully acknowledge the partial financial support of the Army Research Office (DAAH04-95-1-0600, DAAH04-96-10285) and the National Science Foundation (NSF-ECS95-31485)

is often necessary to use multiple models of the robot's behaviour and to devise an appropriate switching rule for deciding which model and/or controller of the system to use. This approach leads in a natural manner to viewing each subsystem in the hybrid system as a switched dynamical system whose timed behaviour can be readily modeled using a timed automaton. Once such an automaton has been extracted, it can be used in conjunction with existing automatic verification tools to analyze the entire system's behaviour.

This paper presents a method for extracting a timed automaton model of this switched dynamical system. The remainder of this paper is organized as follows. Section 2 defines notation. Section 3 introduces the basic switching control system to be considered in this paper. Section 4 summarizes some recent results in bounded amplitude control of such switched systems. Section 5 discusses the use of these results in the extraction of timed automaton models. Section 6 discusses the application of these results for the bounded-amplitude stability of infinitely switched model reference systems. Section 7 summarizes the current status and future directions of this work.

2 Mathematical Preliminaries:

Definition 1. For a finite constant $T > 0$, the finite-horizon infinity norm of a signal $f : \mathbb{R}^+ \rightarrow \mathbb{R}^n$ is defined as

$$\|f\|_{\infty, [0, T]} := \sup_{t \in [0, T]} \|f(t)\|$$

where $\|\cdot\|$ denotes the Euclidean l_2 vector norm. The linear space $\mathcal{L}_{\infty}^n[0, T]$ is defined by

$$\mathcal{L}_{\infty}^n[0, T] := \{f : \mathbb{R}^+ \rightarrow \mathbb{R}^n \mid \|f\|_{\infty, [0, T]} < \infty\}$$

The subset $\{f : \mathbb{R}^+ \rightarrow \mathbb{R}^n \mid \|f\|_{\infty, [0, T]} \leq 1\} \subset \mathcal{L}_{\infty}^n[0, T]$ is denoted $\mathcal{BL}_{\infty}^n[0, T]$.

The infinite-horizon infinity norm of a signal $f : \mathbb{R}^+ \rightarrow \mathbb{R}^n$ is defined as

$$\|f\|_{\infty, [0, \infty)} := \sup_{t \in [0, \infty)} \|f(t)\|$$

where $\|\cdot\|$ denotes the Euclidean l_2 vector norm. The linear space $\mathcal{L}_{\infty}^n[0, \infty)$ is defined by

$$\mathcal{L}_{\infty}^n[0, \infty) := \{f : \mathbb{R}^+ \rightarrow \mathbb{R}^n \mid \|f\|_{\infty, [0, \infty)} < \infty\}$$

The subset $\{f : \mathbb{R}^+ \rightarrow \mathbb{R}^n \mid \|f\|_{\infty, [0, \infty)} \leq 1\} \subset \mathcal{L}_{\infty}^n[0, \infty)$ is denoted $\mathcal{BL}_{\infty}^n[0, \infty)$.

The spaces $\mathcal{L}_{\infty}^n[0, \infty)$ and $\mathcal{BL}_{\infty}^n[0, \infty)$ will often be denoted, respectively, $\mathcal{L}_{\infty}^n$ and $\mathcal{BL}_{\infty}^n$.

Definition 2 (Parameter Variation Set). Given a compact subset $\tilde{\Theta} \subset \mathbb{R}^s$, the parameter variation set $\mathcal{F}_{\tilde{\Theta}}$ denotes the set of all continuous functions mapping $\mathbb{R}^+$ into $\tilde{\Theta}$. For a finite $T > 0$ and a compact subset $\Theta \subseteq \tilde{\Theta}$, the set $\mathcal{F}_{\Theta}[0, T]$ denotes the set of all continuous functions $\theta \in \mathcal{F}_{\tilde{\Theta}}$ which map $[0, T]$ into Θ .

Definition 3 (Linear Parameter Varying (LPV) System). *Given a compact set $\tilde{\Theta} \subset \mathbb{R}^s$, and continuous functions $A : \mathbb{R}^s \rightarrow \mathbb{R}^{n \times n}$, $B : \mathbb{R}^s \rightarrow \mathbb{R}^{n \times n_w}$, $C : \mathbb{R}^s \rightarrow \mathbb{R}^{n_z \times n}$, and $D : \mathbb{R}^s \rightarrow \mathbb{R}^{n_z \times n_w}$, an n^{th} order linear parameter varying (LPV) system is a dynamical system whose dynamics evolve as*

$$\begin{bmatrix} \dot{x}(t) \\ z(t) \end{bmatrix} = \begin{bmatrix} A(\theta(t)) & B(\theta(t)) \\ C(\theta(t)) & D(\theta(t)) \end{bmatrix} \begin{bmatrix} x(t) \\ w(t) \end{bmatrix} \quad (1)$$

where $\theta \in \mathcal{F}_{\tilde{\Theta}}$. The LPV system of definition 3 will be denoted by $\Sigma(\tilde{\Theta}, A, B, C, D)$.

Finally, a positive definite matrix $P \in \mathbb{R}^{n \times n}$ which satisfies

$$A'P + PA + \alpha_1 P + \frac{1}{\alpha_2} PB'BP \leq 0 \quad (2)$$

for $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n_u \times n}$, and scalars α_1 and α_2 , will be denoted

$$P \in \text{FeasRic}(A, B, \alpha_1, \alpha_2)$$

3 Switched Multiple Agent Systems

In this section we model the plant as a model reference system. We're interested in the design and characterization of switched linear controllers which satisfy bounded amplitude performance measures.

Recent work in switched agent control primarily concerns stabilization of linear, uncertain plants. In [10], linear control agents are employed and switched into feedback according to a prediction of controller performance. That work reveals insight into the behaviour of switching systems ($\mathcal{L}_2$ performance) and introduces the concept of *dwell time*, the time that a given control agent must remain in feedback with the plant between switches. Nonlinear control agents are used for bounded amplitude performance in [13]. That work also explores the effects of switching transients and demonstrates that bounds on switching transients can be derived in certain cases. The methods presented in [3] provide sufficient conditions for the bounded amplitude performance of switched linear parameter varying systems with a finite number of switches. The principle results from this paper provide important insight into methods for estimating switching times and hence can be used to extract a timed automaton abstraction of the continuous-state plant.

We assume that the system to be controlled has a differential equation $\dot{x}_p = f_p(x_p, u)$. The desired control task is to move the plant state x_p along a known reference trajectory which is the solution to the $\dot{x}_m = f_m(x_m)$. We will use a multiple agent controller of the form

$$u = K(m, x, t) \quad (3)$$

$$m = \mathcal{D}(x, u, t) \quad (4)$$

where $K(m, x, t)$ is assumed to be an LTI system and $\mathcal{D}(x, u, t)$ is the switching logic used to select which controller will be placed into feedback. The objective is to design the family of controllers $K(m, x, t)$ and the switching logic $\mathcal{D}(x, u, t)$ such that the amplitude of a performance variable $z = \Gamma(x, u, t)$ is bounded above by a constant γ . In other words, we design to satisfy the following constraint,

$$\sup_{t \in [0, T]} \|\Gamma(x, u, t)\| \leq \gamma \quad (5)$$

One method for solving this problem is to use recently developed methods in the synthesis of bounded amplitude linear parameter varying control systems [4] [3]. In this method, we linearize the plant about its reference model and then convert it to a linear parameter varying system. The linearization has the form

$$\dot{x} = A_x(x_m, x, u)x + B_{ux}(x_m, x, u)u + B_{wx}(x_m, x, u) \quad (6)$$

where

$$A_x(x_m, x, u) = \frac{\partial}{\partial x_p} f_p(x_m, 0)$$

$$B_{ux}(x_m, x, u) = -\frac{\partial}{\partial u} f_p(x_m, 0) \quad (7)$$

$$B_{wx}(x_m, x, u) = f_p(x_m) - f_p(x_p, u) - \frac{\partial}{\partial x_p} f_p(x_m, 0)x + \frac{\partial}{\partial u} f_p(x_m, 0)u \quad (8)$$

In this method, once the plant has been linearized, the nonlinear error term, B_{wx} , is added to the linear model as a disturbance input vector.

The next step is to transform the linearized error system into an LPV system. By grouping the (x_m, x, u) -dependent terms in the elements of A_x , B_{ux} , and B_{wx} into real parameters $\theta_1, \theta_2, \dots, \theta_s$, the linearized error model may be written as an LPV system. This grouping forms a map $S : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}^s$ which is called the *parameter mapping*. For piecewise continuous bounded trajectories $x_m(t)$, $x(t)$, and $u(t)$ the parameter mapping S produces a piecewise continuous trajectory $\theta(t) = S(x_m(t), x(t), u(t))$ which is called the *parameter trajectory*. The *nominal parameter trajectory* $\theta_{nom}(t) := S(x_m(t), 0, 0)$. The nominal parameter trajectory clearly represents the parameter trajectory associated with the reference model state trajectory. Deviations from the nominal parameter trajectory may thus be treated as perturbations to the nominal system. To help quantify the size of these deviations, assume that S is a continuous mapping. In particular, assume that S is bounded so that there exists a known constant $k_{zi} > 0$ such that for any $x_m \in \mathbb{R}^n$, $x_1, x_2 \in \mathbb{R}^n$, and $u_1, u_2 \in \mathbb{R}^{n_u}$

$$|S_i(x_m, x_1, u_1) - S_i(x_m, x_2, u_2)| \leq k_{zi} \|z_1 - z_2\| \quad (9)$$

where $\|\cdot\|$ denotes the Euclidean vector norm, $S_i(\cdot, \cdot, \cdot)$ is the i th element of the parameter vector and $z_j = C_1 x_j + D_{12} u_j$ for $j = 1, 2$.

The LPV error system can now be formally defined. Given performance weights C_1 and D_{12} and the initial assumptions listed above, let $\tilde{\Theta} \subset \mathbb{R}^s$ be a compact parameter set such that $\theta_{nom}(t) \in \mathcal{F}_{\tilde{\Theta}}$ and S is the parameter mapping. Consider the linearized error system given in equation 6 and let $A : \mathbb{R}^s \rightarrow \mathbb{R}^{n \times n}$, $B_u : \mathbb{R}^s \rightarrow \mathbb{R}^{n \times n_u}$, and $B_w : \mathbb{R}^s \rightarrow \mathbb{R}^{n \times n_w}$ be defined so that $A_x = A \circ S$, $B_{ux} = B_u \circ S$, and $B_{wx} = B_w \circ S$. The *LPV error system* is the dynamical system

$$\dot{x}(t) = A(\theta(t))x(t) + B_w(\theta(t))w(t) + B_u(\theta(t))u(t) \quad (10)$$

$$z(t) = C_1x(t) + D_{12}u(t) \quad (11)$$

$$y(t) = x(t) \quad (12)$$

for $\theta = S(x_m, x, u) \in \mathcal{F}_{\tilde{\Theta}}$. where

$$B(\theta(t)) = \begin{bmatrix} B_w(\theta(t)) & B_u(\theta(t)) \end{bmatrix},$$

$$C(\theta(t)) = \begin{bmatrix} C_1 \\ I \end{bmatrix} \quad \text{and} \quad D(\theta(t)) = \begin{bmatrix} 0 & D_{12} \\ 0 & 0 \end{bmatrix}$$

This system will often be denoted $\Sigma(\tilde{\Theta}, A, B, C, D)$. The exogenous input $w(t) \equiv 1$ is called the *fictitious disturbance*.

Specific parts of the switched LPV error system are now described. **Control Agents:** Consider the LPV error system $\Sigma(\tilde{\Theta}, A, B, C, D)$. Consider a sequence of times, $\{t_1, t_2, \dots, t_M\}$ and let $\mathcal{I}_{\mathcal{K}} = \{1, \dots, M\}$. The parameter vectors obtained by sampling the nominal parameter trajectory at times t_i for $i \in \mathcal{I}_{\mathcal{K}}$ form a finite collection of *design points*. In other words,

$$\text{Design Points} = \{\theta : \theta = \theta_{nom}(t_i), i \in \mathcal{I}_{\mathcal{K}}\} \quad (13)$$

The i th design point will be denoted as $\theta_{nom}^{(i)}$. With each $i \in \mathcal{I}_{\mathcal{K}}$, associate a *control agent* designed for the LPV error system when the parameter is fixed at the design point $\theta_{nom}^{(i)}$. The i th control agent will be represented by the matrix $K^{(i)} \in \mathbb{R}^{n_u \times n}$ so that, as assumed in section earlier, the control agents are all linear state feedback controllers. The collection of control agents will be denoted as $\mathcal{K}$.

Switching Sets: Switching between the different control agents in $\mathcal{K}$ will be controlled by the parameter vector, θ . In particular, associate with each element of $\mathcal{K}$ a compact subset of the parameter set $\tilde{\Theta}$. This set will be called the *switching set*; the i th switching set associated with control agent $K^{(i)}$ will be denoted as $\Theta_i \subset \tilde{\Theta}$. While this set can be chosen in many ways, attention in this paper is confined to switching sets of the form

$$\Theta_i := \left\{ \theta \mid \sup_j \left| \theta_j - \theta_{nom,j}^{(i)} \right| \leq \vartheta_{out} \right\} \quad (14)$$

where θ_j and $\theta_{nom,j}^{(i)}$ denote the j th components of the parameter vectors θ and $\theta_{nom}^{(i)}$, respectively, and ϑ_{out} is a parameter quantifying the size of the switching set. ϑ_{out} will also be referred to as the *switching parameter*.

Nearest-Neighbor Switching Rule: For a given collection of control agents $\mathcal{K}$ with associated switching sets, there are a variety of *switching rules* which can be invoked. In this paper attention is focused on a *nearest neighbor* switching rule. Given the switching parameter ϑ_{out} , a collection of control agents $\mathcal{K}$, and a collection of parameter sets $\mathcal{C} = \{\Theta_i\}$ as defined in equation 14, suppose that control agent $K^{(i)}$ is in the feedback loop at time t_0 and assume that $\theta(t_0) \in \Theta_i$. Then the control agent $K^{(i)}$ will remain in the feedback loop until the earliest time t_s when the parameter trajectory $\theta(t)$ satisfies $\sup_j \left| \theta_j(t_s) - \theta_{nom,j}^{(i)} \right| = \vartheta_{out}$. At time t_s , the control agent $K^{(m)}$ is then switched into the feedback loop where $m = \arg \min_{j \in \mathcal{I}_{\mathcal{K}}} \left\| \theta(t_s) - \theta_{nom}^{(j)} \right\|$. A parameter trajectory will be said to be *legal* if and only if $\theta(t) \in \cup_{i \in \mathcal{I}_{\mathcal{K}}} \Theta_i$ for all t and $\theta(t)$ is continuous except possibly at times t_s when $\theta(t_s)$ lies on the boundary of a switching set Θ . In particular, a legal parameter trajectory is denoted $\theta \in \mathcal{F}_{\mathcal{C}}$.

From the preceding discussion it is clear a given control agent, $K^{(i)}$, is switched out of the feedback loop when the parameter $\theta(t)$ leaves the switching set Θ_i . In order to guarantee performance properties, it would be advantageous if the parameter $\theta(t_s^+)$ immediately after the switch will also be in the switching set Θ_j . To help guarantee this property, it will be assumed that for all t there exists $l \in \mathcal{I}_{\mathcal{K}}$ such that $\left| \theta_{nom,i}(t) - \theta_{nom,i}^{(l)} \right| \leq \vartheta_{s,i}$ where $\vartheta_{s,i} \leq \vartheta_{out} - k_{zi}\gamma$ for $s = 1, \dots, s$. In other words, this is an assumption that the reference trajectory has been sampled “adequately”.

4 Switching Results

This section summarizes results which we have proven in [3]. Only the results are stated here.

Proposition 1. (LPV Switching Lemma) *Consider any finite constants $r \in (0, 1]$ and $\gamma > 0$, compact sets $\Theta_1, \Theta_2 \subset \tilde{\Theta}$, continuous matrix mappings $A_i : \mathbb{R}^s \rightarrow \mathbb{R}^{n \times n}$ and $B_i : \mathbb{R}^s \rightarrow \mathbb{R}^{n \times n_u}$ for $i = 1, 2$ and constant matrices $C_i \in \mathbb{R}^{n_z \times n}$ for $i = 1, 2$. Let $\mathcal{C} = \{\Theta_1, \Theta_2\}$.*

Suppose there exist constants $\alpha > 0$, $\beta > 0$, and $\rho > 0$ and positive definite matrices P_1 and P_2 such that

$$rP_2 \leq P_1 \tag{15}$$

$$\gamma^2 P_1 \geq C_1' C_1 \tag{16}$$

$$\gamma^2 P_2 \geq C_2' C_2 \tag{17}$$

$$P_1 \in \text{FeasRic} \left(A_1(\theta), B_1(\theta), 2\beta + \frac{\alpha}{r}, \alpha \right) \quad \forall \theta \in \Theta_1 \tag{18}$$

$$P_2 \in \text{FeasRic} (A_2(\theta), B_2(\theta), \rho, \rho) \quad \forall \theta \in \Theta_2 \tag{19}$$

Let w , x , and z be the input, state, and output, respectively, of the dynamical system

$$\begin{bmatrix} \dot{x}(t) \\ z(t) \end{bmatrix} = \begin{cases} \begin{bmatrix} A_1(\theta(t)) & B_1(\theta(t)) \\ C_1 & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ w(t) \end{bmatrix}, & t \in [0, t_s) \\ \begin{bmatrix} A_2(\theta(t)) & B_2(\theta(t)) \\ C_2 & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ w(t) \end{bmatrix}, & t \in (t_s, T] \end{cases} \quad (20)$$

where $t_s > 0$, $T \in (t_s, \infty)$, $\theta \in \mathcal{F}_C$ (i.e. $\theta(t)$ is legal) If $x'(0)P_1x(0) \leq 1$ and $w \in \mathcal{BL}_{\infty}^{n_w}$, then for any switching time satisfying

$$t_s > t_d := -\frac{1}{2\beta} \log r \quad (21)$$

with parameter trajectory $\theta(t) \in \mathcal{F}_C$ satisfying $\theta(0) \in \Theta_1$ and $\theta(t_s^+) \in \Theta_2$,

$$\|z\|_{\infty, [0, T]} \leq \gamma.$$

The LPV switching lemma states three sufficient conditions for bounded amplitude performance of switch LPV error systems.

1. The first condition is that the initial error state, $x(0)$, lie in the ellipsoid $\{x|x'P_1x \leq 1\}$. From theorem 1 of [4], equations 16 and 18 are sufficient to guarantee that this ellipsoid is invariant for times prior to t_s and that any point $\tilde{x} \in \{x|x'P_1x \leq 1\}$ will also satisfy $\|C_1\tilde{x}\| \leq \gamma$. Thus, the first condition guarantees performance on the interval $[0, t_s]$ and is sufficient to ensure performance over the interval $[0, T]$ if no switch were to occur.
2. The second condition is that the parameter trajectories must evolve over the switching sets Θ_1 and Θ_2 so that $\theta(t) \in \mathcal{F}_C$ with the added restriction that $\theta(0) \in \Theta_1$ and $\theta(t_s^+) \in \Theta_2$. This added restriction guarantees that while the system dynamics correspond to (A_i, B_i, C_i) , $i = 1, 2$, the parameter trajectory lies in the set Θ_i ; this condition ensures that the Riccati inequalities of the lemma are valid for the system. The condition of the lemma states that immediately after the discontinuity, the parameter trajectory must satisfy $\theta(t_s^+) \in \Theta_2$ as shown.
3. The final condition is that the switching time must satisfy a *dwell-time* requirement, $t_s \geq t_d$. This guarantees that the state error has had sufficient time to decay so that any transient associated with the switch will not violate performance constraints. At the time of the switch, the invariant ellipsoid associated with the system dynamics switches to $\{x|x'P_2x \leq 1\}$. The dwell-time condition guarantees that at time t_s , the state $x(t_s)$ lies on the interior of the new invariant by ensuring that the state has had sufficient time to decay into the ellipsoid $\{x|x'P_1x \leq r\} \subset \{x|x'P_2x \leq 1\}$. This decay time is characterized by the constant β which parameterizes a bounding exponential. The dwell-time is computed by determining when this bounding exponential satisfies $e^{-2\beta t} = r$.

Lemma 1. *Given performance level $\gamma > 0$ and LPV error system $\Sigma(\tilde{\Theta}, A, B, C, D)$ suppose that the control input to the system is given by $u(t) = Kx(t)$ where K is a constant gain matrix. Let $\bar{\theta}$ be a point on the nominal parameter trajectory and let Θ be any compact subset of $\tilde{\Theta}$ containing $\bar{\theta}$. Suppose that there exists a positive definite matrix P and constants $\alpha > 0$, $\beta \geq 0$, and $0 < r \leq 1$ such that*

$$P \geq \frac{1}{\gamma^2}(C_1 + D_{12}K)'(C_1 + D_{12}K) \quad (22)$$

$$P \in \text{FeasRic} \left(A(\theta) + B_u(\theta)K, B_w(\theta), 2\beta + \frac{\alpha}{r}, \alpha \right) \quad \forall \theta \in \Theta \quad (23)$$

For any $T > 0$, if $x'(0)Px(0) \leq 1$ and $w \in \mathcal{BL}_\infty^{n_w}[0, T]$, then any parameter trajectory $\theta \in \mathcal{F}_\Theta[0, T]$ must also satisfy

$$|\theta_i(t) - \theta_{nom,i}(t)| \leq k_{zi}\gamma \max\{\sqrt{r}, e^{-\beta t}\} \quad \text{for } i = 1, 2, \dots, s \text{ and for all } t \in [0, T] \quad (24)$$

Lemma 1 is important because it implies that if the bound on the parameter trajectory is small enough, then the nominal parameter trajectory represents a reasonable approximation to the true parameter trajectory.

Corollary 1. *Suppose that the conditions of lemma 1 are satisfied with adequate sampling and let $T \in [0, \infty)$. Let $\Theta \subset \tilde{\Theta}$ be a switching set. If $x'(0)Px(0) \leq 1$, $w \in \mathcal{BL}_\infty^{n_w}[0, T]$ and a parameter trajectory $\theta \in \mathcal{F}_\Theta[0, T]$ satisfies*

$$\sup_{1 \leq i \leq s} |\theta_i(T) - \bar{\theta}_i| = \vartheta_{out} \quad (25)$$

at time T , then the nominal parameter trajectory at time T satisfies

$$\vartheta_{out} - k_{zi}\gamma \max\{\sqrt{r}, e^{-\beta T}\} \leq |\theta_{nom,i}(T) - \bar{\theta}_i| \leq \vartheta_{out} + k_{zi}\gamma \max\{\sqrt{r}, e^{-\beta T}\} \quad (26)$$

for $i = 1, 2, \dots, s$.

Corollary 1 is useful because it implies that events in the parameter space (e.g. the parameter trajectory crossing the boundary of a switching set) can be predicted with the nominal parameter trajectory. The time at which the parameter trajectory may intersect the boundary of Θ may be approximated by the times at which the nominal parameter trajectory evolves over points near the boundary of Θ .

5 Automaton Extraction

When combined with a specific switching rule such as the nearest neighbor rule, the nominal parameter trajectory can be used to estimate switching times and the results of possible switches.

Proposition 2. *Given a performance level $\gamma > 0$ and an LPV error system $\Sigma(\tilde{\Theta}, A, B, C, D)$, let $\mathcal{K}$ be a set of control agents which forms a control policy with the nearest neighbor switching rule under adequate sampling. Suppose that at time t_0 , the control input to the system is given by $u(t_0) = K^{(l)}x(t_0)$ where $K^{(l)} \in \mathcal{K}$ is a constant gain matrix. Let Θ_l be a switching set and suppose there exist positive definite matrix P and constant $\alpha > 0$, $\beta \geq 0$, and $0 < r \leq 1$ such that*

$$P \geq \frac{1}{\gamma^2}(C_1 + D_{12}K^{(l)})'(C_1 + D_{12}K^{(l)}) \quad (27)$$

$$P \in \text{FeasRic}\left(A(\theta) + B_u(\theta)K^{(l)}, B_w(\theta), 2\beta + \frac{\alpha}{r}, \alpha\right) \quad \forall \theta \in \Theta_l \quad (28)$$

Define the sets

$$\hat{\Theta}_m := \left\{ \theta \mid \theta \in \tilde{\Theta}, \|\theta - \theta_{nom}^{(m)}\| \leq \|\theta_i - \theta_{nom}^{(q)}\|, m, q \in \mathcal{I}_{\mathcal{K}}, q \neq m \right\}$$

and

$$\mathcal{T}^{(l,m)} := \left\{ t \mid \begin{array}{l} |\theta_{nom,i}(t) - \hat{\theta}_i| \leq k_{max,i}(t) \\ \text{for } i = 1, \dots, s \text{ and some } \hat{\theta} \in \hat{\Theta}_m, \text{ and} \\ \vartheta_{out} - k_{max,i}(t) \leq |\theta_{nom,i}(t) - \theta_{nom,i}^{(l)}| \leq \vartheta_{out} + k_{max,i}(t), \\ \text{for some } 1 \leq i \leq s \end{array} \right\}$$

where

$$k_{max,i}(t) := k_{zi}\gamma \max\{\sqrt{r}, e^{-\beta(t-t_0)}\}$$

If $x'(t_0)Px(t_0) \leq 1$, $w \in \mathcal{BL}_{\infty}$, and a parameter trajectory θ is generated by the nearest neighbor switching rule under adequate sampling, then the switch time, t_s , between the l th and m th systems satisfies $t_s \in \mathcal{T}^{(l,m)}$.

Proposition 2 is important because it provides a way of estimating the time interval over which switches between agents will occur. There are two primary components to the construction of a set $\mathcal{T}^{(l,m)}$:

Switching Destinations The set $\hat{\Theta}_m$ represents the set of all parameter vectors $\theta \in \tilde{\Theta}$ which satisfy

$$m = \arg \min_{j \in \mathcal{I}_{\mathcal{K}}} \left\| \theta - \theta_{nom}^{(j)} \right\|;$$

if the parameter trajectory at time t_s lies in $\hat{\Theta}_m$, a switch to control agent m will take place according to the nearest neighbor switching rule. The sets $\hat{\Theta}_m$ may be represented by a set of affine inequality constraints on the parameter vector θ . Note that these constraints may be computed in an off-line fashion with knowledge of the nominal parameter trajectory and the design points. Since the true parameter trajectory can only be estimated by the nominal parameter trajectory (as a consequence of lemma 1), the times for which $\theta(t) \in \hat{\Theta}_m$ can only be estimated. Let the set

$$\bar{\Theta}_t := \{ \theta \mid |\theta_{nom,i}(t) - \theta| \leq k_{max,i}(t), \text{ for all } i = 1, \dots, s \}$$

For each time t , the set $\bar{\Theta}_t$ is a hyperrectangle centered at $\theta_{nom}(t)$ which contains the true parameter trajectory $\theta(t)$. The first requirement for a time t to belong to $\mathcal{T}^{(l,m)}$ is for the intersection $\bar{\Theta}_t \cap \hat{\Theta}_m$ to be nonempty, which is a relatively simple convex feasibility problem.

Switching Times As a consequence of corollary 1, the times for which the nominal parameter trajectory satisfies

$$\vartheta_{out} - k_{max,i}(t) \leq |\theta_{nom,i}(t_s) - \theta_{nom,i}^{(l)}| \leq \vartheta_{out} + k_{max,i}(t)$$

for some $i = 1, \dots, s$, bound the times for which the true parameter trajectory may intersect the boundary of the parameter set Θ_l . This condition represents an added restriction for a time t to belong to $\mathcal{T}^{(l,m)}$. Note that this requirement may also be expressed with a set of affine inequality constraints on the parameter vector θ which may be computed in an off-line fashion.

Given the preceding descriptions, the set $\mathcal{T}^{(l,m)}$ is constructed by finding all times for which the nominal parameter trajectory lies near the switching surface *and* for which the corresponding set $\bar{\Theta}_t$ has a nonempty intersection with $\hat{\Theta}_m$. Theoretically, constructing the sets $\mathcal{T}^{(l,m)}$ requires checking the feasibility of convex constraints for points along the nominal parameter trajectory at every time, t . Practically, constructing the sets $\mathcal{T}^{(l,m)}$ requires checking for feasibility at a large number of points. This can be done efficiently using a binary search of the reference trajectory.

We can now use the above results to extract a timed automaton. Control agent gains, $K^{(l)}$, were synthesized for the biased systems using the techniques presented in [4]. A MATLAB program was written to implement the conditions associated with propositions 2. First, the switch-time sets $\mathcal{T}^{(l,m)}$ were computed according to the conditions of proposition 2. The nominal parameter trajectory was searched to determine possible switching times and the resulting switches. The result of the search is a tree depicting all possible switches and results for a fixed performance level γ and switching parameter ϑ_{out} .

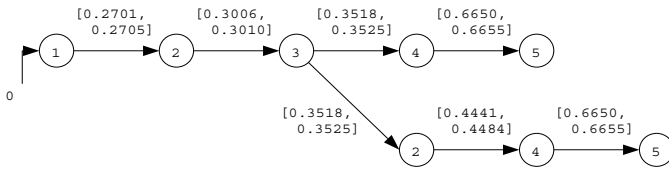


Fig. 1. Performance Validation Tree: $\vartheta_{out} = 1.5$, $\gamma = 0.17$

Using the switching times, $\mathcal{T}^{(l,m)}$, computed for the controlled system (as depicted in figure 1), one may construct a timed automaton [1] to represent the behaviour of the switched system. The timed automaton corresponding to the tree in figure 1 is shown in figure 2; it represents an abstraction of the multiple

agent controlled system obtainable with estimated of switching times, the sets $\mathcal{T}^{(l,m)}$ of lemma 2.

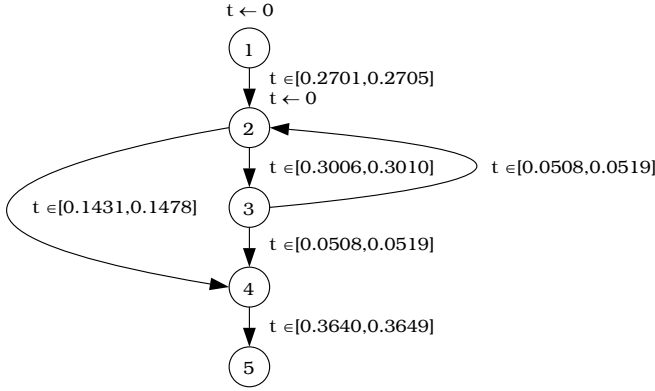


Fig. 2. Timed Automaton for Multiple Agent System: $\vartheta_{out} = 1.5$, $\gamma = 0.17$.

6 Bounded Amplitude Stability for Infinitely Switched Systems

In the presence of bounded disturbances, it is important that we ensure the output of the system is also bounded. We therefore present the notion of bounded-amplitude stability (b.a.s.). This is essentially a requirement for input-output stability.

Definition 4. Let $\gamma > 0$ be a finite real number. Consider a system H described by the differential equations

$$\dot{x} = f(x, w) \quad (29)$$

$$z = h(x, w) \quad (30)$$

where $f : \mathbb{R}^{n \times n_w} \rightarrow \mathbb{R}^n$ and $h : \mathbb{R}^{n \times n_w} \rightarrow \mathbb{R}^{n_z}$ are piecewise continuous bounded mappings. H is said to be bounded amplitude stable (b.a.s.) if and only if for any $w \in \mathcal{BL}_{\infty}^{n_w}[0, \infty)$, $\|z\|_{\infty, [0, \infty)} = \|Hw\|_{\infty, [0, \infty)} \leq \gamma$.

We modify the control agents presented in the preceding section so they have an integrator in their forward loop. The introduction of the integrator ensures that the parameter trajectory, $\theta(t)$, will be continuous, so that the legal parameter switching lemma is no longer needed in the LPV switching lemma. The i th control agent will therefore be represented by the system

$$\dot{u} = v \quad (31)$$

$$v = K_1^{(i)} x + K_2^{(i)} u \quad (32)$$

where $K_1^{(i)}$ and $K_2^{(i)}$ are constant gain matrices of appropriate dimensions. The collection of control agents will be denoted as $\mathcal{K}$.

For analysis and synthesis purposes, the integrator in the control agent will be incorporated into the plant. The modified LPV error system is therefore given by

$$\dot{\hat{x}}(t) = \hat{A}(\theta(t))\hat{x}(t) + \tilde{B}_w(\theta(t))w(t) + \tilde{B}_v v(t) \quad (33)$$

$$z(t) = \tilde{C}\hat{x}(t) \quad (34)$$

where

$$\tilde{A}(\theta(t)) = \begin{bmatrix} A(\theta(t)) & B_u(\theta(t)) \\ 0 & 0 \end{bmatrix}, \quad (35)$$

$$\tilde{B}_w(\theta(t)) = \begin{bmatrix} B_w(\theta(t)) \\ 0 \end{bmatrix}, \tilde{B}_v = \begin{bmatrix} 0 \\ I \end{bmatrix}, \quad (36)$$

$$\tilde{C} = [C_1 \ D_{12}] \quad (37)$$

and $\hat{x} = [x' \ u']$. Consistent with this, we denote the fact that the integrator control agent has a controller $K^{(i)} = \begin{bmatrix} K_1^{(i)} & K_2^{(i)} \end{bmatrix}$.

We can now compute controllers for the given setpoints in the same way as was done above. Using the fact that the integral control agent generates a continuous parameter trajectory it can be readily shown that the legal jump condition is no longer needed.

Note that in many systems, the reference trajectory $\theta_m(t)$ may be a periodic signal. In this case, the continued operation of our system will result in an infinitely switched model reference control system. In this case, we're interested in the *stability* of the infinitely switched system.

Let $\mathcal{T}_d^{(l,m)}$ denote the set $\mathcal{T}^{(l,m)}$ computed with $k_{max,i}(t_d^{(l)})$. Since the dwell-times are fixed, so are the sets $\mathcal{T}_d^{(l,m)}$. Using this information, a timed automaton may be formed representing the switching behaviour of the controlled system. One may then analyze the transitions of the automaton using the LPV switching lemma. If the transitions are separated by the appropriate dwell time, then the switched system is bounded amplitude stable or rather it exhibits bounded amplitude performance even for switching sequences of infinite duration. The following proposition states this result.

Proposition 3. *Given a performance level $\gamma > 0$ and modified LPV error system $\Sigma(\tilde{\Theta}, \tilde{A}, \tilde{B}, \tilde{C}, 0)$ let $\mathcal{K}$ be a set of control agents which form a control policy with the nearest neighbor switching rule under the assumption of adequate sampling.*

Suppose that for each $l \in \mathcal{I}_K$, there exists a positive definite matrix $P^{(l)}$ and constants $\alpha_1^{(l)} \geq \alpha_2^{(l)} > 0$ such that

$$P^{(l)} \geq \frac{1}{\gamma^2} \tilde{C}' \tilde{C} \quad (38)$$

$$P^{(l)} \in \text{FeasRic} \left(\tilde{A}(\theta) + \tilde{B}_v K^{(l)}, \tilde{B}_w(\theta), \alpha_1^{(l)}, \alpha_2^{(l)} \right), \quad \forall \theta \in \Theta_l \quad (39)$$

If, for all possible switching sequences, $k \rightarrow l \rightarrow m$, there exists $r^{(l,m)}$ satisfying

$$r^{(l,m)} \in \left\{ r : rP^{(m)} \leq P^{(l)}, \frac{\alpha_2^{(l)}}{\alpha_1^{(l)}} < r \leq 1 \right\} \quad (40)$$

and

$$\frac{r^{(l,m)}}{\alpha_2^{(l)} - r^{(l,m)}} \log(r^{(l,m)}) \leq \min \mathcal{T}_d^{(l,m)} - \max \mathcal{T}_d^{(k,l)} \quad (41)$$

then the controlled system is b.a.s.

In recent years there has been considerable interest in the stability of switched systems. Results presented in [6] and [7] indicate that the existence of a set of multiple Lyapunov-like functionals is sufficient for asymptotic stability in switched systems. Related work pertaining to linear time invariant switched systems will be found in [11]. The determination of these Lyapunov-like functions, however, is non-trivial. For switched linear time invariant (LTI) systems whose switching sets form cones in the state space, linear matrix inequalities (LMI) [5] were simulatenously suggested in [12] and [8] as a means of computing such Lyapunov-like functions. It should be noted that the conditions obtained above for bounded-amplitude stability are similar to LMI conditions which have been previously reported. In our case, however, we solve a set of LMI's associated with ensuring bounded amplitude stability of the switched system, rather than asymptotic stability.

7 Summary

This paper has outlined a method for the extraction of timed automaton used in the symbolic model checking approach to hybrid system verification. While work on automaton extraction for switched LTI controllers is well understood. It is obvious that nonlinear controllers may be more useful in certain applications. Our current research efforts are focusing on the design of gain-scheduled output feedback controllers. This work is near completion and initial results will be reported at the HS'97 workshop.

References

1. R. Alur and D.L. Dill. A theory of timed automata. *Theoretical and Computer Science*, 126:183–235, 1994.
2. R. Alur, T. Henzinger, and P-H Ho. Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering*, 22:181–201, 1996.
3. C.J. Bett and M.D. Lemmon. Bounded amplitude control using multiple linear agents. Technical Report ISIS-97-004, Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN, 1997.

4. C.J. Bett and M.D. Lemmon. Finite horizon bounded amplitude control of linear parameter varying systems. Technical Report ISIS-97-002, Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN, 1997. Submitted for publication in *Automatica*. Short version to appear 1997 Automatic Control Conference.
5. Stephen Boyd, Laurent El Gaoi, Eric Feron, and Venkataramanan Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*. Society for Industrial and Applied Mathematics, 1994.
6. M. Branicky. Stability of switched and hybrid systems. In *Proceedings of the 33rd Conference on Decision and Control*, pages 3498–3503, Lake Buena Vista, FL, December 1994. IEEE.
7. L. Hou, A.N. Michel, and H. Ye. Stability analysis of switched systems. In *Proceedings of the 35th Conference on Decision and Control*, pages 1208–1212, Kobe, Japan, December 1996. IEEE.
8. M. Johansson and A. Rantzer. Computation of piecewise quadratic lyapunov functions for hybrid systems. Technical report, Lund Institute of Technology, Dept. of Automatic Control, 1996. preprint.
9. Kenneth McMillan. *Symbolic Model Checking*. Kluwer Academic Pub., Norwell, Massachusetts, 1993.
10. A. S. Morse. Supervisory control of families of linear set-point controllers - part 1: Exact matching. *IEEE Transactions on Automatic Control*, 41(10):1413–1431, October 1996.
11. P. Peleties and R. DeCarlo. Asymptotic stability of m -switched systems using lyapunov-like functions. In *Proceedings of the American Control Conference*, pages 1679–1684, Boston, MA, June 1991. IEEE.
12. S. Pettersson and B. Lennartson. Stability and robustness for hybrid systems. In *Proceedings of the 35th Conference on Decision and Control*, Kobe, Japan, December 1996. IEEE.
13. Kameshwar Poolla and Jeff. S. Shamma. Optimal asymptotic robust performance via nonlinear controllers. *International Journal of Control*, 62(6):1367–1389, December 1995.

Sliding Mode Model Semantics and Simulation for Hybrid Systems

Pieter J. Mosterman^{1*}, Feng Zhao^{2**}, and Gautam Biswas^{3***}

¹ Institute of Robotics and System Dynamics,
DLR Oberpfaffenhofen,
D-82230 Wessling, Germany
`Pieter.J.Mosterman@dlr.de`

² Department of Computer and Information Science,
The Ohio State University, Columbus, OH 43210, USA[†]
`zhao@parc.xerox.com`

³ Department of Computer Science,
Vanderbilt University, Nashville, TN 37235, USA
`biswas@vuse.vanderbilt.edu`

Abstract. We describe model semantics and develop a simulation algorithm for characterizing a class of dynamic physical systems operating in the so-called *sliding regimes*. Complex continuous system behavior combines effects that occur at multiple temporal and spatial scales. Behavior generation is simplified by creating system models that employ time scale and parameter abstraction techniques. The resultant hybrid systems exhibit discrete and continuous behaviors, which manifest as piecewise continuous behaviors interspersed with discontinuous changes between the continuous operating modes. Mode transitions are induced by internal state changes and external control signals. Sometimes hybrid systems exhibit *chattering* behaviors at the discontinuous transition boundaries. This presents computational challenges to conventional numerical simulation methods. We develop an efficient, adaptive algorithm for simulating this class of systems, based on a careful analysis of the model semantics at the discontinuous boundaries. Simulation results show that the algorithm is more efficient and accurate for sliding mode systems than conventional integration methods.

1 Introduction

Complex physical systems exhibit behaviors that occur at multiple temporal and spatial scales. To simplify behavior analysis, system models incorporate abstractions that (i) ignore small parasitic effects, and (ii) compress fast behaviors to

* Supported by a grant from the DFG Schwerpunktprogramm KONDISK.

** Supported in part by ONR grant N00014-97-1-0599, NSF NYI grant CCR-9457802, and Sloan Research Fellowship.

*** Supported by grants from PNC, Japan and Hewlett-Packard, Labs, Palo Alto.

[†] Current contact address: Xerox Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, CA 94304, USA

occur at a point in time. The resultant model behaviors are *piecewise continuous*, i.e., they include modes of continuous operation with discrete transition between modes. Building these *hybrid system* models is a difficult task [10]. In previous work [8,15], we have developed a compositional modeling approach with local switching functions to develop hybrid models of physical systems. Dynamic mode switching is attained by a meta-level control model operating on top of the data flow model. The control model is usually a discrete-event system that composes together model fragments to define behaviors in different modes of operation. Examples of hybrid systems include traffic control systems, electric power circuits, reprographic machines, embedded manufacturing processes, and economic models.

Building numerical simulators for these systems is often hampered by the fact that steep gradients occur because of fast nonlinear phenomena. Hybrid models abstract away fast nonlinear effects by invoking discrete mode change functions when system state variables reach or exceed threshold values. The threshold functions can be looked upon as switching surfaces in phase space along which discontinuous changes in the system may occur.¹ Large errors may be avoided in traditional integration schemes like the Runge-Kutta method by using very small time steps around points of discontinuity where steep gradients may occur, at the expenses of incurring significant computational overhead.

The ability to accurately analyze hybrid systems while retaining computational efficiency requires:

- well-defined semantics for defining model components and transition functions, and
- simulation schemes that can seamlessly combine continuous behavior generation methods with discrete mode switching schemes.

This paper focuses on modeling semantics and simulation schemes for a class of hybrid systems operating in the so-called *sliding regimes* where the system chatters between two modes of operation. An example of such a system is the anti-lock braking system employed in automobiles. The switching surfaces in the physical system behavior description arise from modeling artifacts that abstract away the hysteresis effects of small, unmodeled parameters. We develop a simulation algorithm based on Filippov's construction of equivalence in dynamics in sliding regimes [3]. This simulation algorithm is known to be more efficient than conventional schemes that use a fixed step size in behavior generation.

The sliding mode simulation algorithm is based on the observation that the equivalent dynamics on a sliding surface correspond to the limiting behavior when switching tends to be infinitely fast. In previous work [17], we have developed an adaptive algorithm that accurately follows a sliding trajectory and generates control signals at discrete times by exploiting the equivalence in control signals. In contrast, the algorithm described in this paper exploits equivalent dynamics for sliding mode systems and presents an alternate method for adaptively following trajectories at discontinuous boundaries.

¹ A phase space of a dynamic system is a multi-dimensional space defined by the independent state variables of the system.

2 Examples of Hybrid Systems

A typical example of a hybrid system, shown in Fig. 1, is the evaporator vessel in the secondary cooling loop of a fast breeder nuclear reactor [7]. In this system, a sodium pump maintains a sufficient flow of coolant in the loop. To keep the level of fluid in the evaporator vessel at or below a pre-specified maximum, an overflow mechanism is activated to drain excessive fluid once this level is attained. As more fluid comes into the evaporator, the flow through the overflow pipe builds up momentum. The evaporator behavior, described by the level of fluid in the evaporator, L , and the fluid momentum, p , is illustrated for a fixed inflow as a continuous phase space diagram, shown in Fig. 2(a) for a given inflow. At and around the overflow level, L_{th} , the continuous but nonlinear behavior is governed by adhesive forces in the fluid, the intake area of the overflow pipe, and the fluid surface area. In spite of the nonlinearities the system attains a steady state behavior (see Fig. 2(a)). To simplify analysis, the detailed nonlinear effects can be abstracted away, and the overflow mechanism is modeled as having two distinct modes of operation: (i) when the fluid level is below the overflow level, and (ii) when the fluid level is above the overflow level. Behavior in each of the modes can be simplified to an almost linear trajectory (see Fig. 2(b)). This behavior abstraction produces simpler piecewise continuous behaviors with a discrete switch from one behavior to another at the point in time when $L = L_{th}$, which defines the *switching surface*. We refer to this form of model abstraction as a *parameter* abstraction [8].

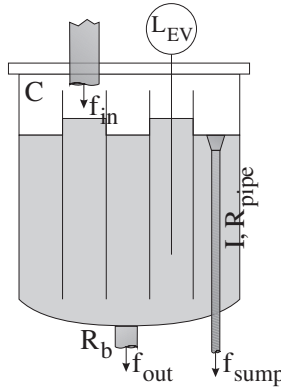


Fig. 1. A hydraulic system with two distinct modes of operation.

Behavior can also be abstracted in time. *Time scale* abstraction compresses the effects of fast change to occur at a point in time. The resulting behavior may become discontinuous [8]. An example is the bouncing ball shown in Fig. 3. A falling ball hits a floor with negative velocity, i.e., $v_{ball} < 0$. The point of contact, where the vertical position of the ball $x_{ball} = 0$, defines the switching

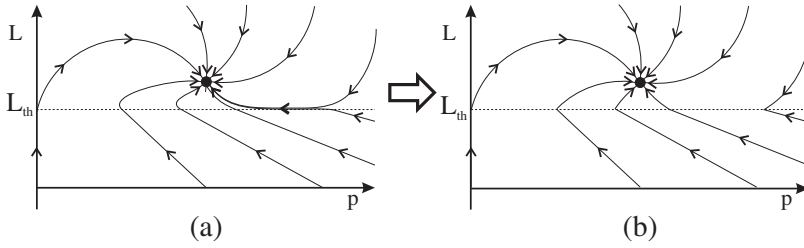


Fig. 2. Phase space of evaporator behavior: (a) Continuous model, (b) Piecewise simpler model.

surface. At this point, fast phenomena occur that transform the kinetic energy of the ball into stored elastic compression energy in the ball, and this is again turned back into kinetic energy as the ball decompresses, and the ball velocity v_{ball} reverses. A phase space plot, x_{ball} versus v_{ball} , of the continuous behavior is shown on the left of Fig. 4. The discontinuous change introduced by time scale abstraction is shown on the right in Fig. 4. The velocity of the ball v_{ball} undergoes an instantaneous jump from $-v_{ball}$ to v_{ball} . This is shown by the double arrow heads in Fig. 4.

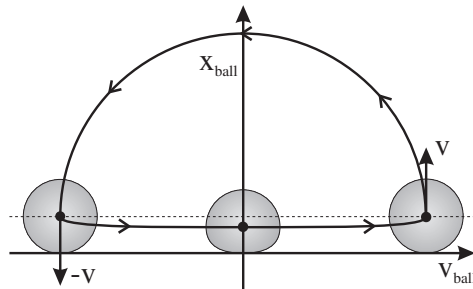


Fig. 3. A bouncing ball.

Introducing parameter and time scale abstractions into physical system models results in piecewise continuous behaviors. Behavior switching occurs at points in time on switching surfaces, and this often leads to jumps in state variable values. The mixture of continuous and discrete behaviors collectively defines the notion of *hybrid* models [4,8,10] of physical systems, represented by an integrated formalism that combines continuous differential equation models with discrete switching functions implemented as Petri nets or finite state automata [1]. This paper develops a simulation algorithm for accurately approximating behaviors of a class of dynamic systems near switching boundaries that result from parameter abstractions.

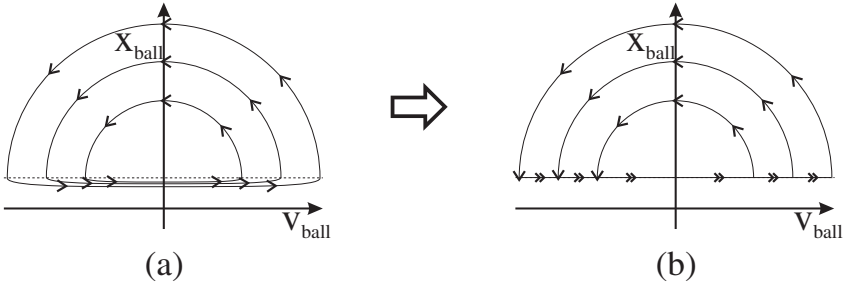


Fig. 4. Phase space of a bouncing ball: (a) Continuous system, and (b) Instantaneous mode change.

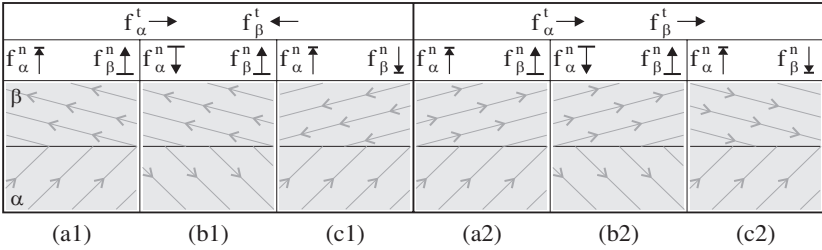


Fig. 5. Types of phase space behaviors near a switching surface: (a) The vector field approaches the surface from one side and leaves from the other side; (b) Vector fields on both sides point away from the surface; (c) Vector fields on both sides point towards the surface. The figure shows these three configurations for two different combinations of tangential components of the vector fields.

At a discontinuous boundary when model switching occurs, a wide variety of behaviors may be observed. A classification presented in [16] and illustrated in Fig. 5, categorizes three types of behaviors. They are characterized by the directions of normal components of the vector fields, f_α^n and f_β^n in modes α and β , respectively. Discrete-time simulators are prone to make large errors at switching boundaries in situations shown on the left in Fig. 6. A way of minimizing these errors is to reduce the simulation time step in the vicinity of the switching boundary so the cross over point may be estimated more accurately. A discontinuity surface can be accurately located by successive halving of time steps, commonly referred to as the *time-step halving method* [11].

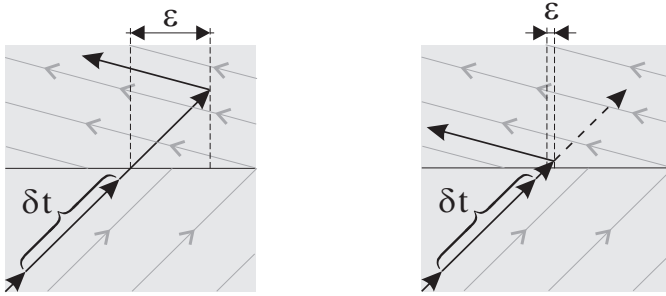


Fig. 6. Errors without time-step adjustment near discontinuities (left) and with time-step halving (right).

Chattering occurs when field vectors in adjacent modes of operation are directed toward the switching surface (Fig. 7). In some cases, such as anti-lock braking systems [12,14], chattering may be an intentional effect. In *sliding mode* operation, the system switches between modes at a very fast rate on the switching surface, producing a fast chattering behavior. This causes the simulated aggregate behavior along the switching surface to progress slowly in time. Chattering limits the step sizes of numerical integration for accuracy reasons. Variable step-size methods are not applicable here since these methods rely on continuous properties of the system variables. To reduce the computational complexity due to the slow progression along the sliding surface, the system dynamics along the surface can be approximated by two methods: (i) equivalence in control [17], and (ii) equivalence in dynamics [3]. In either case, a larger step size can be employed by the integration scheme without introducing intolerable errors.

3 Chattering in Physical Systems

We study the physical origins of chattering using two characteristic multi-mode systems in order to derive a consistent model semantics for simulation: (i) the secondary cooling loop evaporator vessel, and (ii) a cam-follower mechanism.

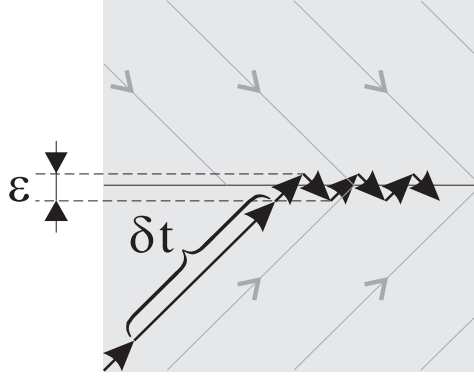


Fig. 7. A hybrid system may chatter.

3.1 The Evaporator

Consider the hydraulics of the evaporator vessel in Fig. 1. There is a constant inflow of liquid into the tank, f_{in} , and an outflow, f_{out} , that depends on the pressure in the tank and the Bernoulli resistance, R_b . As discussed, the system can be modeled to operate in two distinct modes of operation: (i) mode α , where there is no overflow, and (ii) mode β , the overflow mode. The overflow mechanism activates when the liquid level in the evaporator, L , exceeds the threshold value, L_{th} . This causes a flow f_{sump} through a narrow pipe with resistance, R_{pipe} , and inertia, I . When the overflow mechanism becomes active, the pipe inertia builds up flow momentum, p , till steady state when $f_{in} = f_{out} + f_{sump}$.

$$\alpha : \begin{cases} \dot{p} = -\frac{R_{pipe}}{I}p \\ \dot{L} = -\frac{1}{R_b C}L + \frac{f_{in}}{C} \end{cases} \quad (1)$$

$$\beta : \begin{cases} \dot{p} = -\frac{R_{pipe}}{I}p + L \\ \dot{L} = -\frac{1}{IC}p - \frac{1}{R_b C}L + \frac{f_{in}}{C} \end{cases} \quad (2)$$

Suppose the system is in mode α initially. The inflow causes the tank to start filling, and this in turn causes an outflow through resistance R_b . The outflow through the narrow pipe is zero. If steady state is attained before overflow, the inflow, f_{in} , into the tank equals outflow, f_{out} , of fluid and the level of liquid in the tank L becomes steady. However, if L exceeds the threshold level, L_{th} , an immediate switch to mode β occurs (see Fig. 8). The second outflow path in this mode slows the pace at which the tank fills, and this continues till a new steady state level, s_β , where total outflow equals the inflow. This new steady state liquid level is below what would have been attained had the overflow mechanism not been present.

The behavior is more complicated when the level L_{th} is greater than the steady state level L attained in mode β (Fig. 9, the mode is marked in the

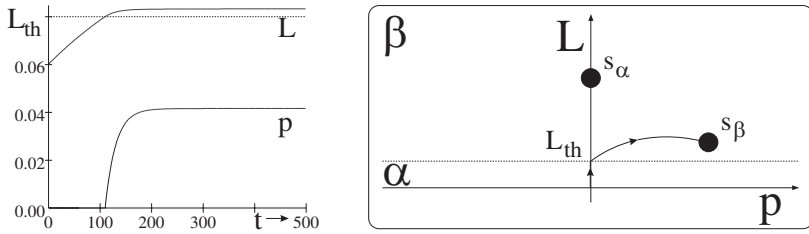


Fig. 8. After an initial transient stage, the evaporator level reaches steady state.

bottom left corner). When the overflow is active in mode β , the system moves towards a steady state with a lower level value. This can cause $L < L_{th}$, which turns off the overflow mechanism, and the system moves back to mode α . The grayed out area in Fig. 9 in each mode represents state vector values that cause a transition to the other mode. The fields in the two modes are directed towards the switching border (L_{th}), and, therefore, independent of the initial conditions. As a consequence, the system can start chattering (see Fig. 10).

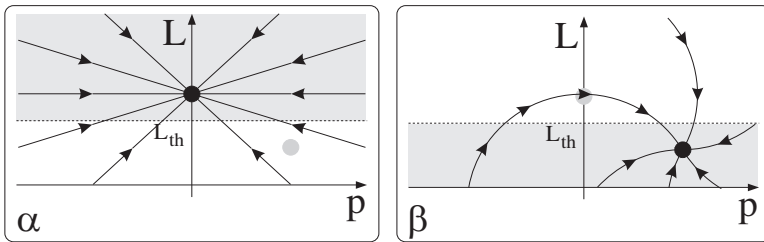


Fig. 9. Phase space of behavior in each mode.

Analysis of the chattering behavior requires a physically consistent treatment of state evolution at L_{th} . Since the domains of the fields for each mode are mutually exclusive, their phase spaces can be combined. Fig. 11 depicts three scenarios by which L_{th} may be approached. In the scenario marked 1, the system approaches L_{th} with a field component in the $-p$ direction in the active mode of operation β . In scenario 2, the behavior path to L_{th} has a 0 component in the p direction in mode β . In scenario 3, the field definitions in modes α and β have equal angles but opposite directions of approach toward L_{th} . The objective is to determine which one of these scenarios defines equilibrium when L_{th} is reached.

To investigate physical behavior, we first observe that in reality the border between the modes of operation is not as crisp as modeled. Modeling abstractions disregard small parameters that are present and affect behavior at the boundary, L_{th} . For example, though small, forces at the rim of the overflow pipe require the fluid level in the tank to be somewhat higher than the rim in order for

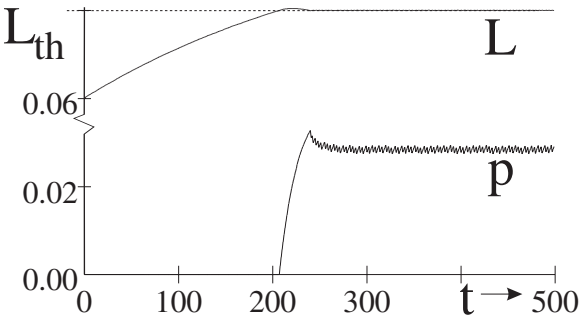


Fig. 10. Chattering between modes with an active and inactive over-flow, $R_b = 1, R_{pipe} = 0.5, I = 0.5, C = 15, f_{in} = 0.25, \Delta T = 0.025$.

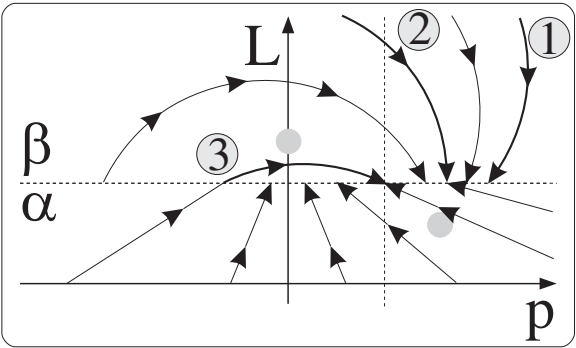


Fig. 11. Concatenation of pieces of phase spaces from modes α and β .

liquid to start pouring in. Therefore, there is a finite time interval in which this excess level is drained, and the overflow mechanism is active. When the level falls below L_{th} the overflow mechanism turns off again. Other higher order physical phenomena, such as cohesive forces in the liquid, result in continuous as opposed to discrete switching behavior around L_{th} .

Other physical effects also contribute to the hysteresis phenomena at the boundary between operational modes, and this can be used to derive correct model semantics for behavior generation at mode boundaries. Fig. 12 shows the effect of a $\pm\epsilon$ hysteresis band around L_{th} . Clearly, the system converges to a recurring point on $L_{th} - \epsilon$ and $L_{th} + \epsilon$ and starts to oscillate between them. At $\lim_{\epsilon \rightarrow 0}$ these recurring points coincide, and the resultant at the common point L_{th} can be computed from the limit values of the field in α and β at this point. If ϵ is small, the curvature of the field in β approaches a straight line. If the direction of the field in α is opposite to the field direction in β at the boundary L_{th} , this point is stable. Trajectory 3 illustrates such a stable point in Fig. 11.

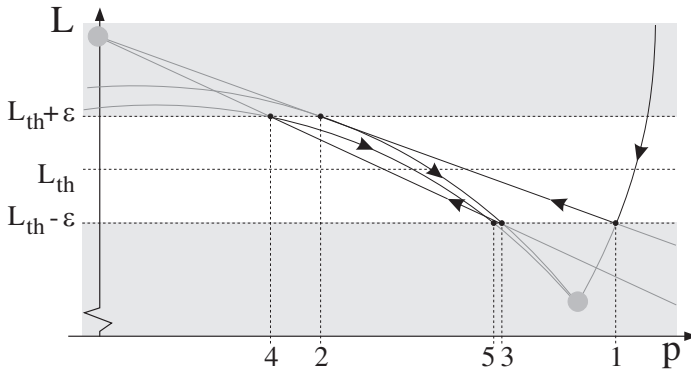


Fig. 12. Iteration across hysteresis effect.

3.2 The Cam-Follower

The cam-follower system in automobiles [5] translates rotational motion into a linear displacement to open and close valves in the engine cylinders (Fig. 13). A spring mechanism ensures contact between the rod and rotating cam but the high rotational velocities (up till several thousands of revolutions per minute) and wear in the spring may result in the rod bouncing on and off the cam, producing a collision phenomena similar to the bouncing ball, where mode switching results in jumps in the values of state variables (Fig. 4). Newton's rule for collisions may be applied using a coefficient of restitution, ϵ , to model loss of energy during collision $v_{rod}^+ - v_{cam}^+ = -\epsilon(v_{rod} - v_{cam})$. Typically, ϵ is a function of impact velocities [2]. If impact is less than a threshold value, $v_{rod} - v_{cam} < v_{th}$, the

collision can be defined to be perfectly non-elastic (i.e., the rod and the cam move together).

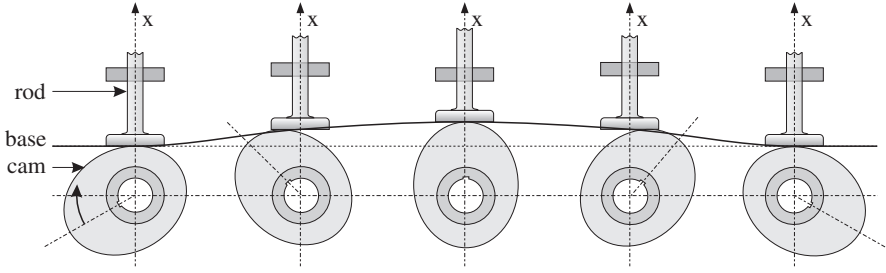


Fig. 13. A cam mechanism opens a valve.

To study the behavior of the cam-follower mechanism in phase space, consider the valve and rod mechanism motion with the valve spring action and the rocker arm friction.

$$\beta : \begin{cases} \dot{v}_{rod} = g - \frac{1}{m}F_{spring} - \frac{R}{m}v_{rod} \\ \dot{F}_{spring} = \frac{1}{C}v_{rod} \end{cases} \quad (3)$$

The behavior of the valve spring and combined inertias, a second order system

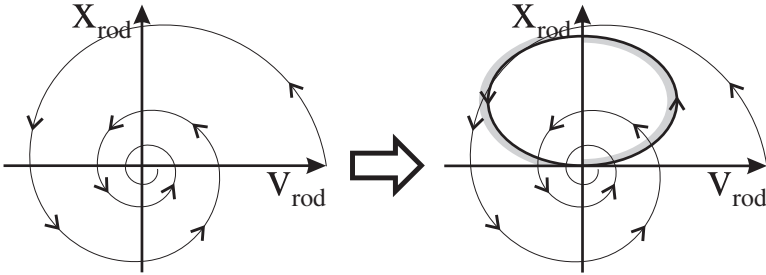


Fig. 14. Phase space of the cam-follower.

with friction, is illustrated in phase space on the left of Fig. 14. The rod velocity oscillates between positive and negative with a decreasing amplitude. The cam mechanism forces the rod velocity to follow an ellipsoid path as shown in the right phase space diagram in Fig. 14.

$$\alpha : \begin{cases} v_{rod} = v_{cam} \\ \dot{F}_{spring} = \frac{1}{C}v_{rod} \end{cases} \quad (4)$$

When the rod is detached from the cam system, a collision can occur if the rod and cam positions are equal, and $v_{rod} < v_{cam}$. If the collision is perfectly non-elastic, the rod velocity instantaneously equals the cam velocity, as is indicated

by the grayed out areas of the phase space in Fig. 14. The rod disconnects from the cam if its deceleration is larger, and this corresponds to the steeper curve in the left half-plane (see Fig. 15).

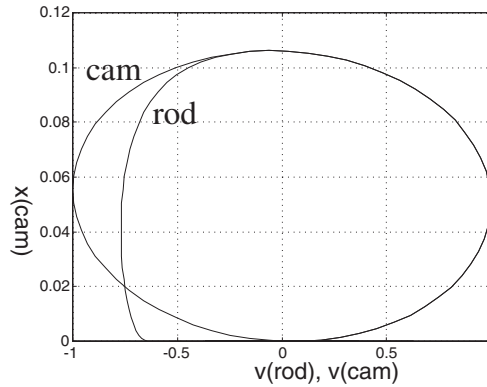


Fig. 15. The rod may disconnect, $R = 10$, $m = 0.5$, $C = 0.02$, $\Delta T = 0.01$.

Simulation runs using numerical approximations of the rod parameters indicate that the collision phenomena described above does occur (see Fig. 16), and the rod begins to exhibit a phenomena where it goes through a sequence of connects and disconnects, i.e., it begins chattering along the switching surface $F_{normal} = 0$ and $v_{rod} = v_{cam}$ as shown on the right in Fig. 16. Like the evaporator, chattering is an artifact of the simulation caused by parameter abstraction in the model. In reality, the rod's elasticity and adhesive forces between the rod and cam surfaces generate higher order behaviors that ensure that the connection remains for a short while before the two disconnect. In the limit, as the values of these parameters tend to 0, the system exhibits sliding behavior along the $v_{rod} = v_{cam}$ surface.

3.3 Model Semantics

When a system operates in the sliding regime, system dynamics in the vicinity of the sliding surface does not appear to be continuous. We adopt the notion of *equivalence in dynamics* for the sliding regime developed by Filippov [3,14], and compare it against an alternative approach, *equivalence in control* [17].

Equivalence in Dynamics Consider the switching surface as an infinitesimal band rather than a crisp border. Equivalent dynamics on the surface is defined as the behavior in the limit as the width of the band tends to zero. This construction preserves the physical meaning of the dynamics at the discontinuous boundaries. Furthermore, it serves as a basis for algorithmically determining the direction and magnitude of the sliding motion.

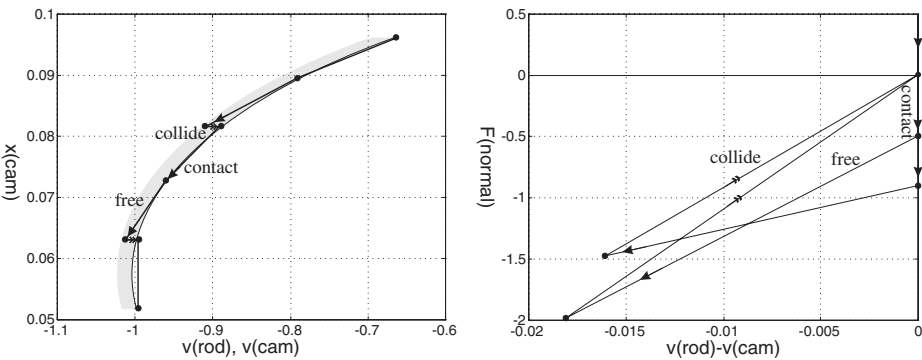


Fig. 16. Numerical simulation may result in chattering, $C = 0.01$.

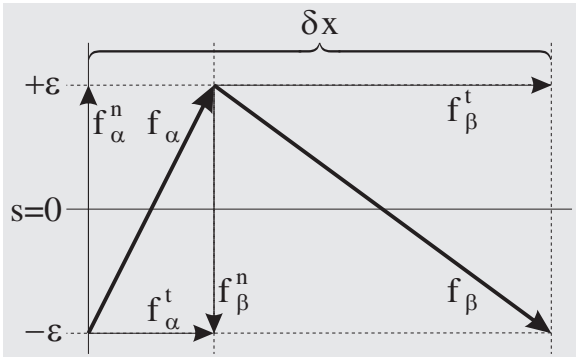


Fig. 17. Motion along a sliding surface.

Assume ϵ is the thickness of the hysteresis band around the sliding surface (see Fig. 17).² If ϵ is small, the fields f_α and f_β on either side of the surface can be represented by their point vector representations with normal components f_α^n and f_β^n , and tangential components f_α^t and f_β^t . The direction of movement is along the sliding surface, and we need to calculate the average velocity on the surface.

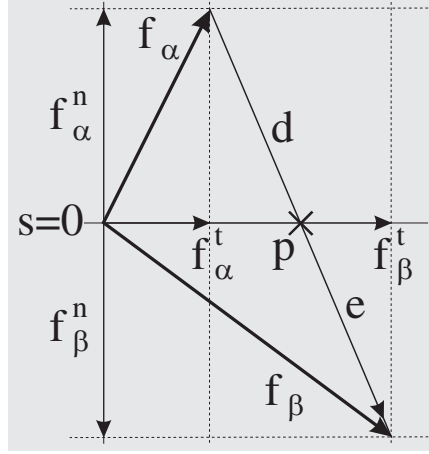


Fig. 18. Filippov construction of sliding motion direction and magnitude.

The time taken by the system to cross the ϵ band is $\delta t_\alpha = \frac{\epsilon}{f_\alpha^n}$ in mode α , and $\delta t_\beta = \frac{\epsilon}{f_\beta^n}$ in mode β . The tangential distance the system travels over two adjacent time intervals ($\delta t_\alpha + \delta t_\beta$) is:

$$\delta x = f_\alpha^t \delta t_\alpha + f_\beta^t \delta t_\beta. \quad (5)$$

We compute the average velocity of the motion on the surface as:

$$\begin{aligned} v &= \frac{\delta x}{\delta t_\alpha + \delta t_\beta} \\ &= \left(\frac{f_\alpha^t \epsilon}{f_\alpha^n} + \frac{f_\beta^t \epsilon}{f_\beta^n} \right) / \left(\frac{\epsilon}{f_\alpha^n} + \frac{\epsilon}{f_\beta^n} \right) \\ &= r f_\alpha^t + (1 - r) f_\beta^t, \end{aligned} \quad (6)$$

where $r = \frac{f_\beta^n}{(f_\alpha^n + f_\beta^n)}$. Thus, the vector v is on the line connecting the end points of f_α and f_β , with r and $1 - r$ as its barycentric coordinates (Fig. 18). Let c be the difference vector $f_\beta - f_\alpha$, and p be the intersection of c with the tangent vector. Let p partition c into two segments, d and e . We have $\frac{e}{d} = \frac{f_\beta^n}{f_\alpha^n}$ by triangle

² This ϵ has no direct relation to the coefficient of restitution used previously.

congruence. Thus, we have shown that the barycentric coordinate for p is the same as that for v (recall $\frac{r}{1-r} = \frac{f_\beta^n}{f_\alpha^n}$). This corresponds to Filippov's construction, i.e., the vector v is the same as the tangent vector.

In fact, the formula $v = rf_\alpha + (1-r)f_\beta$ can be used to compute v , where r is determined from the normal components of the two vector fields if the angle between the sliding surface and vector fields is known.

Equivalence in Control Another approach to deriving the dynamics on a sliding surface is the method of equivalent control. For systems linear in control, this is an identical approximation to the equivalent dynamics [13]. However, nonlinear control may derive different behaviors since the true system behavior near the sliding surface can be attributed to hysteresis phenomena. Utkin [13] shows that the method of equivalent dynamics derives sliding behavior closer to the true dynamics than the method of equivalent control in these situations. In some cases, where system behavior is not *well-behaved* near the switching surface, e.g., the case of a saturated high gain amplifier, where system variable values tend to infinity close to the discontinuity, equivalent control may generate better approximations [6]. The reason is that there are no higher order hysteresis effects, therefore, modeling with equivalent dynamics, which assumes hysteresis, results in the generation of deviant behaviors. We also note that equivalent control methods can derive chatter-free, equivalent control signals directly. In fact, the method of [17] exploits this property to obtain chatter-free simulation and control algorithms.

It is critical to analyze the semantics of the discontinuity, and not introduce hysteresis effects in the model if they are not a true governing phenomena. Incorrect analysis may lead to models that are not well-behaved, and, therefore, cannot be correctly analyzed by equivalent dynamics. This issue needs to be investigated in future research.

4 Sliding Mode Simulation Algorithm

The sliding mode algorithm has been implemented as part of the hybrid system simulation engine [9] using a time-step halving forward Euler numerical integration scheme. When a mode switch between fields β and α is detected (see Fig. 19), a binary search is invoked to determine the switching point to a pre-specified accuracy. After the mode switch is executed and the state vector is transferred to mode $x_\alpha(t_s)$, it is checked whether the new mode, α , persists for at least one time step, ΔT . If it does not, the sliding mode simulation algorithm (Algorithm 4) is activated. The first step computes $x_\alpha(t_s + \Delta T)$ based on f_α with $x_\alpha(t_s)$ as the initial point. Discontinuous changes may take place, and the resulting $x_\beta(t_s + \Delta T)$ is computed. $x_\beta(t_s^+)$ is computed from $x_\alpha(t_s)$ and f_α . Again, $x_\beta(t_s^+)$ may differ from $x_\alpha(t_s^+)$ due to jumps in the state vector between α and β . Next, $x_\beta(t_s^+ + \Delta T)$ is computed from $x_\beta(t_s^+)$ and f_β , and jumps between β and α are taken into account when $x_\alpha(t_s^+ + \Delta T)$ is computed. $x_\beta(t_s + \Delta T)$

is taken as the initial point of the vector $x_\alpha(t_s^+ + \Delta T) - x_\beta(t_s + \Delta T)$. A binary search is performed along this vector given a pre-specified number of steps to determine $x(t_s + \Delta T)$ on the sliding surface. After $x(t_s + \Delta T)$ and the corresponding field are determined, checks are made to make sure chattering does not persist, and behavior continues to evolve in time.

Algorithm 1. Sliding Mode Simulation

Require: A mode switch has occurred from f_β to f_α
 Compute $x_\alpha(t_s + \Delta T)$ from $x_\alpha(t_s)$ and $f_\alpha(t_s)$
 Infer f_β and $x_\beta(t_s + \Delta T)$ {A discontinuous change between $x_\alpha(t_s + \Delta T)$ and $x_\beta(t_s + \Delta T)$ may occur.}
 Compute $x_\alpha(t_s + \epsilon)$ from $x_\alpha(t_s)$ and $f_\alpha(t_s)$
 Infer f_β and $x_\beta(t_s + \epsilon)$ from $x_\alpha(t_s + \epsilon)$
 Compute $x_\beta(t_s + \epsilon + \Delta T)$ from $x_\beta(t_s + \epsilon)$ and $f_\beta(t_s + \epsilon)$
 Infer f_α and $x_\alpha(t_s + \epsilon + \Delta T)$
while $f_\alpha \neq f_\beta$ **do**
 Compute $x_d = x_\alpha(t_s + \epsilon + \Delta T) - x_\beta(t_s + \Delta T)$
 $g = \frac{1}{2}$, $g_{accuracy} = \frac{1}{4}$
 for a given number of iterations **do**
 $x_s = x_\beta(t_s + \Delta T) + g * x_d$
 Infer f_γ from x_s
 if $f_\gamma = f_\alpha$ **then** $g = g + g_{accuracy}$ **else** $g = g - g_{accuracy}$
 $g_{accuracy} = \frac{1}{2}g_{accuracy}$
 end for
 $t_s = t_s + \Delta T$
 $x_\alpha(t_s) = x_s$
 if $f_\alpha \neq f_\gamma$ **then** $f_\alpha = f_\beta$
 Compute $x_\alpha(t_s + \Delta T)$ from $x_\alpha(t_s)$ and $f_\alpha(t_s)$
 Infer f_β and $x_\beta(t_s + \Delta T)$
 Compute $x_\alpha(t_s + \epsilon)$ from $x_\alpha(t_s)$ and $f_\alpha(t_s)$
 Infer f_β and $x_\beta(t_s + \epsilon)$ from $x_\alpha(t_s + \epsilon)$
 Compute $x_\beta(t_s + \epsilon + \Delta T)$ from $x_\beta(t_s + \epsilon)$ and $f_\beta(t_s + \epsilon)$
 Infer f_α and $x_\alpha(t_s + \epsilon + \Delta T)$
end while

4.1 The Evaporator

Consider the chattering behavior in Fig. 10 for the evaporator shown in Fig. 1. Simulation with time-step halving and sliding mode algorithm produces the results shown in Fig. 20. The temporal, dynamic behavior of the system is no different but the error in the sliding mode is greatly reduced. This is further clarified in the phase space plots in Fig. 21. A fixed-step simulation produces a large error (same order of magnitude as the chattering) at $p = 0$ in determining when the overflow mechanism becomes active, i.e., when $L > 0.8$. However, when time-step halving is used to accurately determine the switching surface, it reduces the integration step to the lower bound when chattering occurs. The system behavior is then determined using the sliding mode algorithm.

4.2 The Cam-Follower System

The results of numerical simulation of the cam-follower system is shown in Fig. 16. On the plot on the right in Fig. 16, the negative plane of the $v_{rod} - v_{cam}$

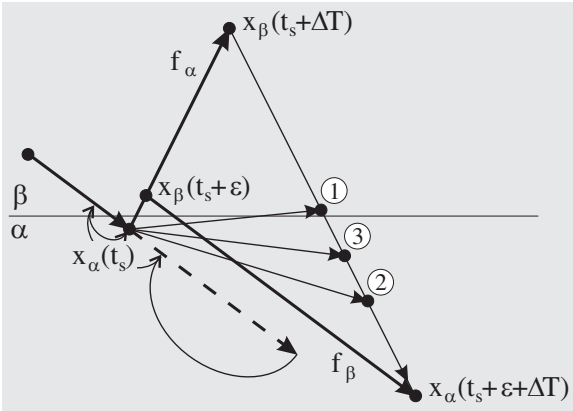


Fig. 19. The sliding mode numerical simulation algorithm.

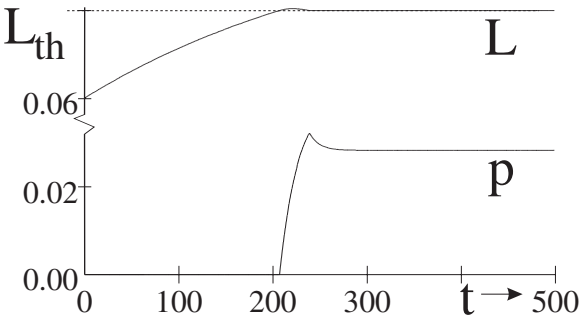


Fig. 20. Simulation of the evaporator using the sliding mode algorithm.

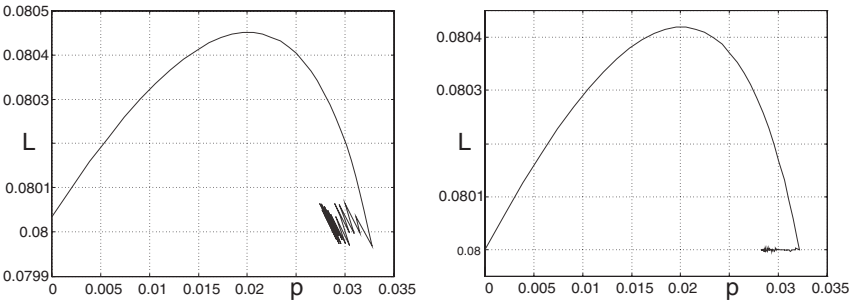


Fig. 21. Fixed step Euler (left) vs. time-step halving Euler with sliding mode simulation (right).

axis results in a non-elastic collision with $v_{rod} = v_{cam}$. This mode α , where the rod and cam system are connected, is active on the $v_{rod} = v_{cam}$ axis as long as F_{normal} is positive or 0. When F_{normal} becomes negative, the system switches to mode β , where the rod is disconnected from the cam and moving freely. For certain parameter values, this moves the system in the plane where a non-elastic collision occurs, and the system moves back to the $v_{rod} = v_{cam}$ axis, with $F_{normal} > 0$. This chattering behavior is an artifact of the numerical time step. For $\Delta T \rightarrow 0$, the system would remain at $(0, 0)$, corresponding to movement along the sliding surface, $v_{rod} = v_{cam}$. Fig. 22 illustrates the operation of the sliding mode algorithm. It shows that in mode β the rod and cam are disconnected and v_{rod} (thick lines) differs from v_{cam} (thin line). The non-elastic collision that follows immediately results in a jump from $x_\beta(t_s + \epsilon + \Delta T)$ to $x_\alpha(t_s + \epsilon + \Delta T)$ which corresponds to $v_{rod} = v_{cam}$. In mode α , the rod and cam are connected, and $v_{rod} = v_{cam}$ also. Therefore, the system slides on the switching surface and there is no error due to chattering. This agrees with the simulation results when higher order physical phenomena, such as adhesive forces between the rod and cam, are incorporated into the system model.

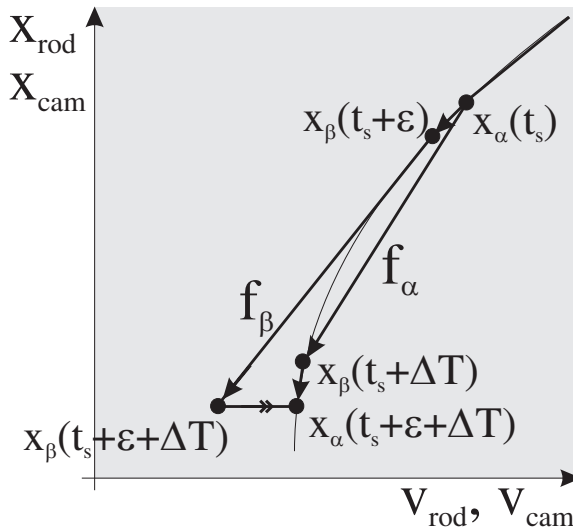


Fig. 22. Sliding mode simulation for the cam-follower system.

Simulation results using a fixed step Euler function produce the chatter error shown on the left plot in Fig. 23. The chatter error can be reduced by using a smaller step size, but time-step halving would increase the computational complexity significantly because the step size would be reduced to its lower bound very quickly during simulation. Sliding mode simulation accounts for the discontinuous jump in rod velocity. As shown on the right in Fig. 23, this produces an error-free behavior. Simulation results also show that the sliding mode simu-

lation correctly turns off when system behavior moves away from the switching surface.

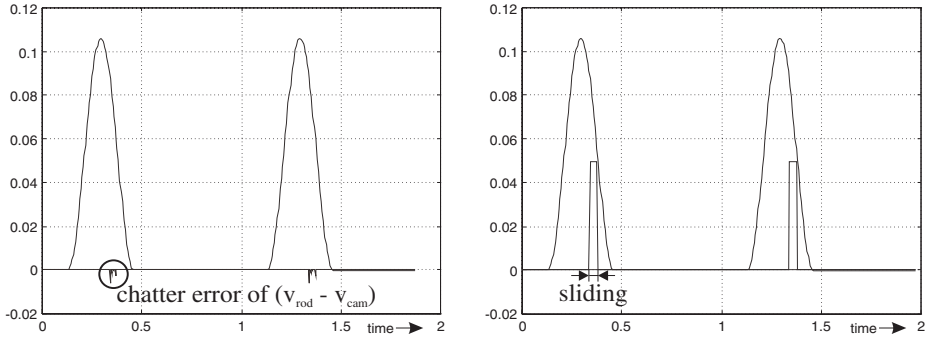


Fig. 23. Sliding mode simulation during an interval of time.

5 Conclusions

Modeling and analysis of hybrid systems present distinct challenges to researchers. Of particular interest are the development of numerical methods that generate accurate approximations of system behavior, especially at and around mode transition boundaries where discontinuous changes can occur in system variables. To accommodate the steep slopes that can arise at the boundaries, simulation algorithms have to reduce step size to estimate slopes more accurately and generate more accurate behaviors. This paper extends our original hybrid system simulation algorithm to accommodate an interesting class of behaviors, the sliding mode, that can occur at mode change boundaries.

Sliding mode systems move along sliding surfaces because of continuous interaction between two adjacent, alternating operating modes. However, modeling abstractions may lead to small, higher order dynamic effects such as parasitic inertia, capacitance, and resistance not being included in the systems model. Both the modeling abstractions and the discrete-time simulation can introduce chattering along the surface where system behavior seems to switch between two modes of behavior at a very fast rate. As discussed, simulation errors can be kept small, by reducing the step size small in numerical integration so that fast chattering motions are not missed. The result is that simulated time progresses only in small increments, and the slower, sliding movement along the switching surface is not simulated efficiently.

Based on a physical model semantics, we have developed a sliding mode simulation algorithm using the Filippov equivalence criteria applied to dynamic behavior generation. Our implementation has shown that the simulation maintains consistency of the temporal behavior along the sliding surface while keeping

the error term small. The algorithm has performed well in several engineering applications where discontinuities in physical behavior result in mode switches.

References

1. A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1974.
2. R.M. Brach. *Mechanical Impact Dynamics*. John Wiley and Sons, New York, 1991.
3. A. F. Filippov. Differential equations with discontinuous right-hand sides. *Matematicheskii Sbornik*, 51(1), 1960.
4. J. Guckenheimer and S. Johnson. Planar hybrid systems. In Panos Antsaklis, Wolf Kohn, Anil Nerode, and Shankar Sastry, editors, *Hybrid Systems II*, pp. 202–225. Springer-Verlag, 1995. Lecture Notes in Computer Science, vol. 999.
5. C.W. Ham, E.J. Crane, and W.L. Rogers. *Mechanics of Machinery*. McGraw-Hill Book Company, Inc, New York, New York, 4th edition, 1958.
6. A.E. Mattson. On Object-oriented Modeling of Relays and Sliding Mode Behavior. *Proc. Triennial IFAC World Congress*, vol. F, pp. 259–264, 1996.
7. P.J. Mosterman and G. Biswas. Monitoring, Prediction, and Fault Isolation in Dynamic Physical Systems, *Proc. AAAI-97*, pp. 100–105, Rhode Island, August 1997. AAAI Press, 445 Burgess Drive, Menlo Park, CA, 94025.
8. P.J. Mosterman and G. Biswas. A theory of discontinuities in dynamic physical systems. *Journal of the Franklin Institute: Engineering and Applied Mathematics*, 335B(3), pp. 401–439, 1998.
9. P.J. Mosterman and G. Biswas. Principles for Modeling, Verification, and Simulation of Hybrid Dynamic Systems. in *Proceedings of the Fifth International Conference on Hybrid Systems*, pp. 21–27, Notre Dame, Indiana, September, 1997.
10. A. Nerode and W. Kohn. Models for hybrid systems: automata, topologies, controlability, observability. In *Hybrid Systems I*, Springer-Verlag, 1993. Lecture Notes in Computer Science, vol. 736.
11. T.S. Parker and L.O. Chua. *Practical Numerical Algorithms for Chaotic Systems*. Springer-Verlag, 1989.
12. J. E. Slotine and W. Li. *Applied Nonlinear Control*. Prentice Hall, Englewood Cliffs, NJ, 1991.
13. V.I. Utkin. Equations of the Slipping Regime in Discontinuous Systems, I. *Automation and Remote Control*, 32(12):1897–1907, 1971.
14. V.I. Utkin. *Sliding Modes in Control and Optimization*. Springer-Verlag, 1992.
15. F. Zhao. Designing dynamics using geometric constraints. In *Proceedings of the 14th IMACS World Congress on Computational & Applied Math*, Atlanta, GA, July 1994.
16. F. Zhao. Qualitative reasoning about discontinuous control systems. In *Proceedings of IJCAI-95 Engineering Problems for Qualitative Reasoning Workshop*, pp. 83–93, Montreal, Canada, August 1995.
17. F. Zhao and V.I. Utkin. Adaptive simulation and control of variable-structure control systems in sliding regimes. *Automatica: IFAC Journal*, 32(7):1037–1042, 1996.

Time-Deterministic Hybrid Transition Systems

Simin Nadjm-Tehrani*

Dept. of Computer & Information Science,
Linköping University,
S-581 83 Linköping, Sweden
`simin@ida.liu.se`

Abstract. Hybrid transition systems in their full generality describe continuous behaviour by a set of equations in each mode – an algebraic or differential equation for each *state* variable in terms of inputs and other state variables. Each discrete transition may be taken according to a (non-deterministic) time constraint.

In this paper we restrict this model to time-deterministic discrete transitions. Thus, every transition is guarded by a condition g and has a fixed delay t . Different transitions may have different delays (including zero), but progress is enforced after the delay. Using this restriction and a composition operator which uses union of mode sets we then prove certain compositionality properties. In particular, that the parallel composition of two subsystems produces a system whose semantics is defined in terms of semantics of its constituents provided that it has a run. The restriction is well-justified in a large class of control applications where the complex mode-changing software is realised as a synchronous program.

1 Introduction

Hybrid theories combine the theory for continuous dynamic systems with the theory for discrete dynamic systems. While the theoretical work is carried out, safety-critical systems are being built and analysed at great costs. Successful application of formal verification techniques to such systems needs to start from engineering models of hybrid systems and systematically transform them to analysable models in the theory. The work presented here draws from the experience with a number of case-studies, where the idea has been to start from the engineering models of physical systems “as they are”.

Hybrid Transition Systems (HTS) were proposed in 1993 as an attempt to capture both mode-switching physical systems and non-deterministically timed computer systems in a single formalism. The formalism was studied in the context of several realistic examples: a two car no-collision scenario [17], a 16th century siphon pump machine [23,22], and the landing gear system of an aircraft [18,19] among others. Some of these examples were verified by the application of deductive methods. Simpler models were augmented by addition of

* This work was supported by the Esprit LTR project 22703 (SYRF) and the Swedish board for technical research (TFR).

invariances after which they could be model-checked as linear hybrid automata (LHA) [2].

In this paper we propose a restriction of the formalism and prove certain compositionality properties, while keeping the expressive style so that engineering models can still be plugged in with no additional effort. The motivation for the restrictions are twofold. First, a number of changes in the definition of the operational semantics and parallel composition operator facilitates proofs of compositionality. Second, restriction to time-determinism particularly suits analysis of systems controlled by programs from the so-called synchronous family of languages [10], as described below.

Hybrid transition systems in their full generality describe behaviours of systems as interleavings of continuous phases of activity (having positive durations) and discrete transitions (taking zero time). Each discrete transition, however, may be taken according to a (non-deterministic) time constraint. A transition is taken a least l time units and at most u time units after it has been enabled ($0 \leq l, u \leq \infty$).

The work on case-studies suggests the use of time-deterministic control programs in many applications, and the need for deductive reasoning about cases where models of the physical environment is non-trivial (eliminating the possibility of model checking) [9]. We therefore propose to restrict the model to time-deterministic transitions while keeping the expressiveness with regard to dynamic continuous systems. Each discrete transition is thus required to be taken within a fixed period of time from the time at which it was enabled. However, in a composed system, the fixed periods may differ – due to transitions belonging to different subsystems.

Hence, we expect to keep the best of both worlds in the following sense. Hybrid models are often built up from several modules each representing either a control unit or a physical apparatus (mechanics, hydraulics, electronics, etc.). For physical systems, our experience shows that models derived using systematic modelling (e.g. by bond graphs) are naturally described by switching phases of continuous activity, immediately on satisfaction of certain conditions (i.e. $l = u = 0$). For control programs, many realisations have a fixed period for each control function ($l = u = d$ for some period d). Analysing the logical and causal behaviour of such synchronous programs [10] on their own is well-established [6,11]. By deriving the hybrid model we can now extend the analysis to the closed loop system behaviours – including assessment of “the synchrony hypothesis”.

Safety properties of such closed loop systems can sometimes be verified on discrete models, but that requires the derivation of a discrete model of the environment [25]. As synchronous languages are becoming more widespread in industrial systems, and interfaces to verification tools are forthcoming (see the Lustre-PVS connection for example [4]), it is even interesting to formally analyse the timing behaviour of such systems in relation to their environments. This generally requires continuous analysis combined with deductive proofs. The restriction to time-determinism enables the direct translation of a synchronous

program to a time-deterministic HTS without transition delays (length of computation step). Adding transition delays and the model of the environment we get a model for the whole system to reason about.

A different application of the time-deterministic model can be the composition of timed models of synchronous subsystems, each having their own computational period; the resulting system being a timed description of asynchronous networks of synchronous processes. Analysis of such applications is a subject for future work.

The rest of this paper describes certain features of hybrid transition systems for natural modelling of engineering systems in section 2. Then follows the formal definition of time-deterministic HTS, and the parallel composition operator for this class of systems in section 3. Section 4 treats issues related to compositionality and progress properties, and section 6 discusses relation to other works.

2 Hybrid transition systems

Hybrid transition systems treat the discrete and continuous elements in each state on a par. That is, all elements of state are conceptually represented as piecewise continuous functions of time. Each system has a finite number of modes. In each mode, the continuous elements of the state are concretely represented by a set of differential and algebraic equations in state space form (see 1), where a real-valued variable may change either in accordance with a differential equation or according to an algebraic equation in each mode. This facilitates modelling the cases where the dimension of the system is changed from one mode to another (see the siphon pump in [23]). In any case, the representation of continuous change as above provides a natural interface to engineering models which have this form, and which can be plugged into the hybrid model without additional transformation (e.g. addition of invariances and exit conditions as in LHA).

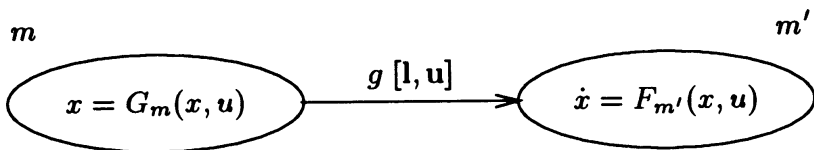


Fig. 1. A schematic illustration of hybrid transition systems as a graph

The discrete elements of state (piecewise constant functions of time) are represented as variables taking constant values in each mode. Thus, all changes in state are captured by differential and algebraic equations. Changes of mode are conditional upon a *guard* becoming true, i.e. a boolean expression g over the state and input variables. The change in input variables is not constrained.

The uniform treatment of continuous and discrete elements of state provides a natural means of communication between subsystems. Instead of shared labels or events we treat every state change as immediately visible by all other subsystems. Thus communication is by shared variables, where each variable is only allowed to be changed by one subsystem (the one which has this variable among its state variables). Other subsystems may (or may not) have this variable as an input variable – a model which supports non-symmetric communication (broadcast as opposed to rendezvous).

The model is specially useful for modular developments. For each component the change in state is defined in terms of current state and the current input. The input is, however, not modelled explicitly (unless it is the state of another subsystem). This allows the proper treatment of disturbances or inputs whose modelling is to be postponed (e.g. the driver behaviours in [17]).

Consider a mode change conditional upon the guard g and having a timing constraint $[l, u]$, as depicted above. Then a watchdog for the guard can result in setting a timer to zero whenever the value of g changes to true. From this point onwards the transition *may* be taken, but at the point when the timer hits u the transition *must* be taken. In what follows, we consider the systems where all the transitions *must* be taken at the point which is exactly t time units apart from the time when their guard becomes true.

3 Time-deterministic HTS

We now define time-deterministic hybrid transition systems (TD-HTS) formally.

Definition 1. *A time-deterministic hybrid transition system is a tuple $\langle M, X, U, F, I, T \rangle$, where*

M : is a non-empty set of mode sets $\{m_1, \dots, m_q\}$. For a simple system m_i are singleton, for a composed system each m_i is a set of mode elements.

X : is a set of typed state variables with disjoint subsets X_c and X_d , $X = X_c \cup X_d$, where the domain of X_c variables is $\mathbb{R}$. The set of states of the system, S , is the set of type consistent interpretations of the variables in X .

U : is a set of typed input variables $U = U_c \cup U_d$ where the domain of the variables in U_c is $\mathbb{R}$ and $X \cap U = \emptyset$.

F : $M \times X \rightarrow \mathcal{E}$ is a function associating an equation with each state variable in each mode, where $e = F(m, x)$ has the following forms:

- if $x \in X_d$ then $e = \lceil x = c \rceil$ for some value c of the right type
- if $x \in X_c$ then
 - $F(m, x) = \lceil \dot{x} = f(\bar{x}, \bar{u}) \rceil$ for some function f of appropriate type,
 - or $F(m, x) = \lceil x = g(\bar{x}, \bar{u}) \rceil$ for some function g of appropriate type
 where $\bar{x}$ and $\bar{u}$ denote vectors of variables over X_c and U_c .

I : is an initial configuration consisting of $\langle m_0, s_0 \rangle$, where $m_0 \in M$ and $s_0 \in S$.

T: is a set of mode transitions $\tau = \langle m, m', g, t \rangle$ where $m, m' \in M$, g is a boolean expression over terms of the form $z \# w$, with $z \in X \cup U$, $\# \in \{=, \leq, \geq, <, >\}$, and w is a type consistent term over X and U , elements from their value domains, and uninterpreted function symbols. $t \in \mathbb{N}$ corresponds to an exact delay on the transition τ .

□

We denote by $\mathcal{I}(x)$ the value of the variable x in the interpretation $\mathcal{I}$. The guard g is defined to be true in an interpretation $\mathcal{I}$ if it evaluates to true when all the variables z in g are substituted by the values given by $\mathcal{I}(z)$.

The operational semantics can be informally described as follows. A run for a system is defined from the initial configuration, for a given input time-function (a function from the reals to interpretations for the input variables). The run consists of a sequence of mode-state-input-time tuples where the changes in the input components are governed by the input time-function at selected time points. The recorded time points are those at which a mode change takes place, or some guard to some transition changes its truth value. The state changes are compatible with the solutions to equations in each mode. The (discrete) mode changes are recorded by having the *same* time component and different mode-state components (though the value of continuous state variables are unchanged at mode changes). Such a mode change appears in a run if the guard g for some transition τ has been true for the duration t as dictated by τ . All enabled transitions take place at the end of their respective durations t (i.e. they *must* take place). If there are several such transitions, then one will appear non-deterministically in the current position in the run, and the others will also take place (with the same time component but at the next position in the run) provided they are still enabled in the current mode.

Next we give a formal account of the operational semantics. In definitions below we represent the interpretation of a set of variables Z as the set $\{\langle z, v \rangle \mid z \in Z\}$, where each v is a value of the right type.

Definition 2. Let $\langle M, X, U, F, I, T \rangle$ be a time-deterministic hybrid transition system, and γ denote a finitely variable function from $\mathbb{R}$ to the set of interpretations over U . Then the **run** of the system with input γ is an infinite sequence of situations $\sigma_0, \sigma_1, \dots$ such that:

- $\sigma_j = \langle m_j, s_j, e_j, t_j \rangle$ where $e_j = \gamma(t_j)$
- $t_0, t_1, \dots$ is a progressive time sequence with $t_j \in \mathbb{R}$
- $\sigma_0 = \langle m_0, s_0, \gamma(0), 0 \rangle$
- each state component $s_j \in S$, and is compatible with the trajectory of the system in mode m_j – i.e. $s_j(x) = \gamma_x(t_j)$ where γ_x denotes the solution to the DAE in mode m_j as defined by F with respect to variable x , given the initial values corresponding to s_i and e_i from the last position i in the run when the mode changed, a position $i < j$, such that $m_i \neq m_j$, and $\forall k \ i < k \leq j \ m_k = m_j$

- at every t_j either the mode changes, or the guard to some transition changes truth value, i.e. $m_j = m_{j+1} \rightarrow \exists \langle m, m', g, t \rangle \in T$ such that g is true (false) in $s_j \cup e_j$ and false (true) in $s_{j+1} \cup e_{j+1}$
- mode changes take no time, i.e. $m_j \neq m_{j+1} \rightarrow t_j = t_{j+1}$
- a transition is taken only if it is enabled for long enough, i.e. $m_j \neq m_{j+1} \rightarrow \exists \tau = \langle m, m', g, t \rangle \in T \exists i \leq j$ such that $\forall i \leq k \leq j$, g is true in $s_k \cup e_k$, $m \cap m_k \neq \emptyset$ and $m' \cap m_{j+1} \neq \emptyset$
- for every transition with guard g and delay t , if g becomes true at a time point t_i and stays true at all subsequent situations prior to the time point $t_j = t_i + t$, then the transition is taken at t_j provided that it can be taken, i.e. $\forall \tau = \langle m, m', g, t \rangle$, if $t_j = t_i + t$, g true in $s_i \cup e_i$, and $\forall i \leq k \leq j$, g true in $s_k \cup e_k$ and $m \cap m_k \neq \emptyset$ then $\exists \sigma = \langle m'', s, e, t_j \rangle$ where $m' \cap m'' \neq \emptyset$, $s(x) = s_j(x)$ for all $x \in X_c$, and is otherwise defined by $F(m'', x)$ (for $x \in X_d$).

□

Note that we want to force progress as a rule. So any transitions which have been enabled “for long enough” *must* be taken. On the other hand, we do not want to force determinism — specially since parallel composition of several subsystems may lead to independent transitions getting ready for being taken at the same time. The semantics thus allows for all such transitions to take place in an arbitrary order but at the same time point *provided* that they are not in causal conflict with each other. The latter is not explicit in the semantic definition but is implicit: a system which does not have a run according to the above definition does not have a well defined semantics. The detection of causal paradoxes, e.g. that a taken transition disables an already enabled transition, may be investigated using similar procedures to those for synchronous systems [6] – taking no regard to the timing delays.

Next we give the definition for composition of TD-HTS. Note that the composition operation is defined provided that a condition on shared variables holds. This might be slightly problematic with certain systems in which common outputs are desired, e.g. subsystems which both emit the same alarm signal in different situations. This situation can however be remedied by renaming of the state variable in one subsystem after composition.

Definition 3. Given time-deterministic HTS $H_1 = \langle M_1, X_1, U_1, F_1, I_1, T_1 \rangle$, and $H_2 = \langle M_2, X_2, U_2, F_2, I_2, T_2 \rangle$, such that $X_1 \cap X_2 = \emptyset$ and $M_1 \cap M_2 = \emptyset$, we define their **composition** denoted by $H_1 \parallel H_2$, as the time-deterministic HTS $H = \langle M, X, U, F, I, T \rangle$ where:

$$M = \{m_1 \cup m_2 \mid m_1 \in M_1, m_2 \in M_2\},$$

$$X = X_1 \cup X_2,$$

$$U = (U_1 \cup U_2) - X,$$

$F : M \times X \rightarrow \mathcal{E}$ where $\mathcal{E}$ is the set of equations $\mathcal{E}_1 \cup \mathcal{E}_2$, such that

$$F(m, x) = F_i(m_i, x), \text{ for } x \in X_i, m_i \subset m \in M, m_i \in M_i$$

$$(i \in \{1, 2\}),$$

T is the smallest set of transitions such that:

if $\langle m_1, m'_1, g_1, t_1 \rangle \in T_1$, then for all $m \in M_2$, we have
 $\langle m_1 \cup m, m'_1 \cup m, g_1, t_1 \rangle \in T$, and
 if $\langle m_2, m'_2, g_2, t_2 \rangle \in T_2$, then for all $m \in M_1$, we have
 $\langle m \cup m_2, m \cup m'_2, g_2, t_2 \rangle \in T$.

□

Proposition 1. *The parallel composition operator is commutative and associative.*

Proof. Follows directly from the definitions and the properties of set union.

□

4 Compositionality of TD-HTS

We are interested in compositionality of hybrid systems in two different ways.

- Proving assertions of the type “if H_1 and H_2 satisfy a particular property, so does $H_1 \parallel H_2$ ”.
- Proving that “if $H_1 \parallel H_2$ satisfies a property P and H_1 is equivalent to H_3 in some sense, then $H_3 \parallel H_2$ satisfies P ”.

The first one is needed for bottom-up modelling and verification of systems, and the second for making simplifications prior to analysis, or for refining abstract designs to implementations. In this paper we treat the first aspect of compositionality. That is, we show that composition of systems preserves certain interesting properties. A central property for hybrid transition systems is having a well-defined semantics in terms of a set of runs. Thus, we would like to show that if H_1 and H_2 each have well-defined semantics in terms of a (non-empty) set of runs, then their composition also has a run. Note that the definition of a TD-HTS does not guarantee existence of a run for the system. In particular, systems with obscure behaviours (e.g. the so-called Zeno behaviours in which time progresses in small increments but never beyond a particular bound) do not have a run, but are not syntactically excluded. Unfortunately, we can not exactly characterize systems whose states are finitely variable (have finite number of discrete state state changes in a finite interval of time), or compositions of systems which have no runs.

Therefore, we proceed as follows. First, we distinguish systems in which an infinite number of discrete transitions are possible in a single point in time. The intuition for this rests on experience with modelling realistic systems. Models of physical systems which are derived with systematic modelling techniques do not exhibit such behaviours, despite the fact that in these systems, structural change can be naturally modelled by an instantaneous transition [23]. Models of computer systems are definitely non-Zeno as long as there are non-zero delays associated with every transition. Therefore, we define the notion of *admissible* systems which at least do not exhibit the undesired behaviour of changing infinitely often in a single time point.

Definition 4. Let $H = \langle M, X, U, F, I, T \rangle$ be a TD-HTS. We associate a labelled graph $G_H = \langle V, E, L \rangle$ with H such $m \in M$ iff $v_m \in V$, and $\langle v, v' \rangle \in E$ iff $\exists \tau = \langle m, m', g, t \rangle \in T$; the labelling function L associates the label (g, t) with the edge $\langle v_m, v'_m \rangle$. We define the TD-HTS H as **admissible** iff for every cycle in G_H , if all labels are of the form $(g, 0)$, then there are at least two edges in the cycle with guards g and g' , where g and g' are mutually exclusive when evaluated in all interpretations of $X \cup U$. □

Proposition 2. If H_1 and H_2 are admissible then $H_1 \parallel H_2$ is admissible. □

Obviously, this property does not guarantee existence of a run when two systems are composed, but it eliminates some cases which depend on inappropriate enabling of guards.

Next we show that the parallel composition of H_1 and H_2 , produces a system H whose semantics is defined in terms of semantics of its constituents provided that it has a run. That is, a sequence belongs to the set of runs of H if it is grounded in runs of H_1 and H_2 in a sense that we make more precise as follows.

Proposition 3. Let H_1 and H_2 be two TD-HTSs. Let γ_1, γ_2 and γ be three input time functions with the following properties:

1. H_i has a run with γ_i , $i \in \{1, 2\}$.
2. $\gamma(t)(u) = \gamma_1(t)(u)$ if $u \in U_1 - X_2$, and $\gamma(t)(u) = \gamma_2(t)(u)$ if $u \in U_2 - X_1$.
3. $\gamma_1(t_j)(u) = e_j(u)$ at every position j of the run for H_2 with γ_2 if $u \in U_1 \cap X_2$, and $\gamma_2(t_j)(u) = e_j(u)$ at every position j of the run for H_1 with γ_1 if $u \in U_2 \cap X_1$.

If $H = H_1 \parallel H_2$ has a run with γ , then for every such run of H , there is some run of H_1 with γ_1 (and H_2 with γ_2) such that for every element at position i in the run of H there is a corresponding element with the same time component t_i in the run of H_1 (or H_2) such that the restriction of the element in H to (state, mode and input) variables of H_1 (or H_2) gives the element in the run for H_1 (or H_2). □

Proof. By induction on the sequence of situations in the runs of H . Consider an arbitrary run of H with γ .

Base step: for the initial state the condition is trivially true.

Induction step: Consider a fragment of the run: $\dots, \sigma_i, \sigma_{i+1}, \dots$

We will show that for any element σ_{i+1} at position $i + 1$ there is an element related to it at a position $j < i + 1$ in the run for H , such that, if there is an element corresponding to σ_j in some run of H_1 with γ_1 (or some run of H_2 with γ_2), then there is an element corresponding to σ_{i+1} in that run of H_1 (or H_2). There are two cases:

1. $m_i \neq m_{i+1}$

According to the operational semantics for H there exists a transition $\tau =$

$\langle m, m', g, t \rangle$ in T such that $m_i \cap m \neq \emptyset, m_{i+1} = m'$, and g has been true in every $s_k \cup e_k$ since a position j in which g became true. Assume that σ_j restricted to variables of H_1 (or H_2) exists in a run for H_1 (or H_2). Then according to the operational semantics, a situation corresponding to σ_{i+1} with the same time component exists in the run of H_1 (or H_2).

2. $m_i = m_{i+1}$

According to the operational semantics there is a transition with guard g in H such that g is false (true) in $s_i \cup e_i$ and true (false) in $s_{i+1} \cup e_{i+1}$. According to the definition of parallel composition this transition (guard) can be traced to one of the two systems H_1 and H_2 . Without loss of generality assume that it comes from H_1 . Consider a run for H_1 with γ_1 . Assume that there is a situation σ_k with time component equal to t_i in this run for H_1 , whose mode and state components are restrictions of m_i and s_i to variables of H_1 . We have to show that σ_{k+1} in that run corresponds to a restriction of σ_{i+1} to variables of H_1 . First, we show that g is false (true) in $s_k \cup e_k$. This easily follows since s_k is a restriction of s_i to X_1 , variables in g are a subset of $X_1 \cup U_1$, and due to conditions 2 and 3 in the antecedents of the proposition, the value of these variables is the same as that in $s_i \cup e_i$.

Next we show that g changes truth value at σ_{k+1} and that takes place at time point t_{i+1} . Again, since the variables in g are a subset of $X_1 \cup U_1$, based on definition of F in parallel composition, and restrictions 2 and 3 in the antecedent to the proposition, we can state the following: if t_{i+1} is the first time point after t_i in which g changed truth value in the run of H , then t_{i+1} is also the first time point after σ_k in the run of H_1 in which g changes truth value. Hence, according to the operational semantics, there exists a situation $\sigma_{k+1} = \langle m', s', e', t_{i+1} \rangle$ in the run of H_1 such that $m' \subset m_i$, $s' = s_{i+1}$ restricted to X_1 , and $e' = \gamma_1(t_{i+1})$.

□

5 Current application

In the SYRF project we are investigating formal verification of a climatic chamber case study provided by Saab Aerospace. The physical environment consists of a chamber with two outlets, a fan to change the flow of air in the chamber, and a heater to warm the incoming air. The objective of the controller is to keep the temperature and flow of air in the chamber within predefined distance from dynamically changing reference values. To ensure that these objectives can be met the system performs continuous regulation as well as monitoring, the latter giving rise to several modes of operation.

The controller is realised as a synchronous program which can be represented in a statechart-like model with different regulation activities associated with different modes. The controller program (or its functional specification) is too large to be included in this paper. The physical environment has so far been modelled with certain simplifying assumptions – nevertheless giving rise to the

following non-linear equation describing the change of chamber temperature in terms of the flow and the input heat power. Here T_{chamb} denotes the inside temperature, T_{in} the incoming air temperature, u_{heat} the voltage applied to the heater, and q the (volume-based) flow of the air in the chamber. The k -terms are constants determined by the range of values involved and the chamber characteristics.

$$\begin{aligned} \dot{T}_{chamb} = & 1/k_{chamb} \\ & (q T_{in} k_{in} + \\ & u_{heat}^2 k_{heat} - \\ & T_{chamb}(q k_{out} - k_{loss})) \end{aligned}$$

Further details of this case study which gives rise to a non-trivial hybrid system can be found in [20]. The physical environment model is modular in the sense that it excludes elements such as events (thresholds) used by the controller and dictated by the requirements specifications.

The initial modelling step using composition of a synchronous program (with fixed delays) and the physical model is facilitated by the TD-HTS model. The next step is transformation of this model to a form which is directly analysable with existing formal verification tools. These are the type of models usually assumed as given in the verification literature. Analysis of the properties of the system and formulation of intermediate invariances is currently in progress.

6 Related works

We share a common aspiration with data flow approaches like Signal [5]: that of treating the discrete and continuous elements on a par.

Hybrid transition systems can be seen as a modular version of phase transition systems [15], our main contribution being the treatment of compositionality and separation of input and state components. Other versions of phase transition systems [16] include the notion of important events which, as well as enabling conditions (corresponding to our guards), include assertions from the requirements specification for a system. Hybrid transition systems do not mix the model of a system and its requirements.

Hybrid automata [2] differ mainly on the model for communication and parallel composition. There, communication is by shared synchronisation labels. Moreover, composition is defined on systems which have the same set of continuous variables, making modular models of different subsystems less natural. Hybrid automata also require invariants in order to force progress. This is often a significant additional modelling step not directly present in engineering models. In our model, progress is by default, and addition of invariances are seen as part of the verification process.

Modular modelling appears in several other frameworks, e.g. in [24,1]. The work by Westhead and Hallam [24] models hybrid computations as a limit to

discrete synchronous computational processes. Abadi and Lamport [1] discuss composition of, and decomposition into, specification modules. Their specifications are representations of safety and liveness properties in temporal logic. Separation of input and state arises naturally in models proposed within control theory, e.g. several models discussed in [7]. However, rather than compositionality, aspects such as stability are in the focus of these discussions.

A hybrid theory which treats aspects of compositionality can be found in the work by Lynch et.al. [14] wherein both communication by shared variables and shared labels is considered. This work also attacks proving non-Zenoness by switching to a game theoretic framework.

Going to the domain of timed systems, there are works which treat composition in presence of non-deterministic timing constraints in variants of timed automata [21,12]. Sifakis and Yovine distinguish between transitions which *must* take place and those which *may* take place by using a notion of deadline in addition to the usual notion of invariance in timed automata. Kesten, Manna and Pnueli, on the other hand, separate the progress conditions and enabling conditions by using different assertions for each. The former being global assertions and the latter as local predicates on each transition.

7 Summary and future works

The paper presents a model for hybrid systems where all elements of state (discrete and continuous) are represented as state variables. Modes of continuous activity are governed by differential and algebraic equations in state space form, and communication is by shared variables. The model is a modification of hybrid transition systems presented earlier — the modifications being restriction to time-determinism and composition by union of mode sets. The restrictions are based on practical experience with modelling realistic engineering systems, and an intention to link to verification efforts on the synchronous family of languages. A treatment of compositionality for open subsystems was discussed and a proof of preservation of semantics on composing subsystems was presented.

Future works include continued investigation of the above industrial case study, and the investigation of how proof techniques from computer science and control theory can be combined in systems having diverse requirements specifications.

References

1. M. Abadi and L. Lamport. Conjoining Specifications. *ACM transactions on Programming Languages and Systems*, 1995.
2. R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The Algorithmic Analysis of Hybrid Systems. *Journal of Theoretical Computer Science*, 138:3–34, 1995.
3. R. Alur, T.A. Henzinger, and E. Sontag, editors. *Proc. of the DIMACS International Workshop on Verification and Control of Hybrid Systems, LNCS 1066*. Springer Verlag, 1996.

4. S. Bensalem, P. Caspi, and C. Parent-Vigouroux. Handling Data-Flow Programs in PVS. Technical report, VERIMAG, 1996. Available through <http://www-verimag.imag.fr//SYNCHRON/SYRF/HTML97/a341.html>.
5. A. Benveniste. Compositional and Uniform Modelling of Hybrid Systems. In Alur et al. [3], pages 41–51.
6. G. Berry. The Foundations of Esterel. In *Proofs, Languages and Interaction: Essays in Honour of Robin Milner*. MIT Press, 1998. To appear.
7. M. S. Branicky. *Studies in Hybrid Systems: Modeling, Analysis, and Control*. PhD thesis, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, June 1995. Dissertation no. LIDS-TH-2304.
8. J.W. de Bakker, C. Huizing, W.P. de Roever, and G. Rozenberg, editors. *Real-Time: Theory in Practice, Proc. REX workshop 1991*. Springer Verlag, LNCS 600, 1992.
9. B. Dutertre and V. Stavridou. Formal Requirements Analysis of an Avionics Control System. *IEEE Transactions on Software Engineering*, 25(5):267–278, May 1997.
10. N. Halbwachs. *Synchronous Programming of Reactive Systems*. Kluwer Academic Publishers, 1993.
11. N. Halbwachs, F. Lagnier, and P. Raymond. Synchronous observers and the verification of reactive systems. In M. Nivat, C. Rattray, T. Rus, and G. Scollo, editors, *Third Int. Conf. on Algebraic Methodology and Software Technology, AMAST'93*, Twente, June 1993. Workshops in Computing, Springer Verlag.
12. Y. Kesten, Z. Manna, and A. Pnueli. Verifying Clocked Transition Systems. In Alur et al. [3], pages 13–40.
13. H. Langmaack, W.-P. de Roever, and J. Vytöpil, editors. *Proc. of the 3rd. International Conference on Formal Techniques in Real-time and Fault-tolerant Systems, LNCS 863*. Springer Verlag, 1994.
14. N. Lynch, R. Segala, F. Vaandrager, and H.B. Weinberg. Hybrid I/O Automata. In Alur et al. [3], pages 496–510.
15. O. Maler, Z. Manna, and A. Pnueli. From Timed to Hybrid Systems. In de Bakker et al. [8], pages 447–484.
16. Z. Manna and A. Pnueli. Models for Reactivity. *Acta Informatica*, 30:609–678, 1993.
17. S. Nadjm-Tehrani and J-E. Strömberg. From Physical modelling to Compositional models of Hybrid Systems. In Langmaack et al. [13], pages 583–604.
18. S. Nadjm-Tehrani and J-E. Strömberg. Proving Dynamic Properties in an Aerospace Application. In *Proc. of the 16th International Symposium on Real-time Systems*, pages 2–10. IEEE Computer Society Press, December 1995.
19. S. Nadjm-Tehrani and J-E. Strömberg. Verification of Dynamic Properties in an Aerospace application. *Formal Methods in System Design*, 1998. To appear.
20. The SYRF Project. Work package 7 Deliverable: Analog/discrete synchronous design – Annex A.7.1a. Available from <http://www-verimag.imag.fr//SYNCHRON/SYRF/HTML97/a71.html>, December 1997.
21. J. Sifakis and S. Yovine. Compositional Specification of Timed Systems. In *Proc. of the 13th annual Symposium on Theoretical Aspects of Computer Science, STACS'96, LNCS 1046*, pages 347–359. Springer Verlag, 1996.
22. J.-E. Strömberg, S. Nadjm-Tehrani, and J. Top. Switched Bond Graphs as Front-end to Formal Verification of Hybrid Systems. In Alur et al. [3], pages 282–293.

23. J.E. Strömberg and S. Nadjm-Tehrani. On Discrete and Hybrid Representation of Hybrid Systems. In *Proceedings of the SCS International Conference on Modeling and Simulation (ESM '94)*, pages 1085–1089, Barcelona, June 1994.
24. M. Westhead and J. Hallam. Modelling Hybrid Systems as the Limit of Discrete Computational Processes. In *Proceedings of the International Conference on Robotics and Automation*. IEEE, 1996.
25. M. Westhead and S. Nadjm-Tehrani. Verification of Embedded Systems using Synchronous Observers. In *Proceedings of the 4th International Conference on Formal Techniques in Real-time and Fault-tolerant Systems, LNCS 1135*, pages 405–419. Springer Verlag, September 1996.

Heuristic Optimization and Dynamical System Safety Verification

Todd W. Neller*

Knowledge Systems Laboratory
Stanford University, Gates Building 2A
Stanford, CA 94305-9020, USA
neller@ksl.stanford.edu

Abstract. Given a heuristic estimate of the relative safety of a hybrid dynamical system trajectory, we transform the initial safety problem for dynamical systems into a global optimization problem. We compare untuned performance of several Simulated Annealing and Multi Level Single Linkage method variants, and discuss the dynamic use of knowledge gained during optimization.

1 Introduction

Given a simulated hybrid dynamical system S , a set of possible initial states I , and a set of “unsafe” states U , we wish to verify nonexistence of an S -trajectory from I to U within t_{max} time units. We call this the *initial safety problem*. Suppose we are given an approximate measure of the relative safety of a trajectory. More specifically, let f be a function taking an initial state i as input, and evaluating the S trajectory from i such that $f(i) = 0$ if and only if the S -trajectory from i enters U within t_{max} time units, and $f(i) > 0$ otherwise. Then verification of the initial safety problem can be transformed into the global optimization (GO) problem:

$$\min_{i \in I} (f(i)) \stackrel{?}{>} 0$$

GO methods may therefore terminate when i is found such that $f(i) = 0$. Given that f does not generally have an analytic form, we do not assume the availability of derivatives. Since each evaluation of f may require a computationally expensive simulation, we are particularly interested in GO methods which perform relatively few evaluations of f . In this context, we compare several variants of Simulated Annealing (SA) and Multi Level Single Linkage (MLSL) methods and assess their suitability for our purposes. We discuss the use of knowledge of f gained in the course of GO, and consider the extent to which some GO methods assume special properties of the local optimization (LO) procedures they use.

* This work was supported by the Defense Advanced Research Projects Agency and the National Institute of Standards and Technology under Cooperative Agreement 70NANB6H0075, “Model-Based Support of Distributed Collaborative Design”.

2 Motivation

Our research was largely motivated by the following safety verification task: Given bounds on the system parameters of a stepper motor (e.g. viscous friction, inertial load), bounds on initial conditions (e.g. angular displacement and velocity), and an open-loop motor acceleration control, verify that no scenario exists in which the motor stalls. We model the motor's continuous dynamics using ODEs given in [1]:

$$\begin{aligned}\dot{\theta} &= \omega \\ \dot{\omega} &= \frac{(-i_a N_b \sin(N\theta) + i_b N_b \cos(N\theta) - D \sin(4N\theta) - F_v \omega - F_c \text{sign}(\omega) - F_g)}{(J_l + J_m)} \\ i_a &= (V_a - i_a R + \omega N_b \sin(N\theta))/L \\ i_b &= (V_b - i_b R - \omega N_b \cos(N\theta))/L\end{aligned}$$

where θ and ω are motor shaft angular displacement and velocity, i_a and i_b are coil A and B current, V_a and V_b are coil A and B voltage, R and L are coil resistance and inductance, N is the number of rotor teeth, N_b is the maximum motor torque per amp, D is the maximum detent torque, F_v is the viscous friction, F_c is the Coulomb friction, F_g is the gravitational torque load, and J_l and J_m are load and motor shaft inertia. For this system we classify a stall as deviation of $\frac{\pi}{N}$ or more radians from the current desired θ equilibrium.

The motor is stepped by reversing polarity of the coil voltages in alternation (see Figure 1).

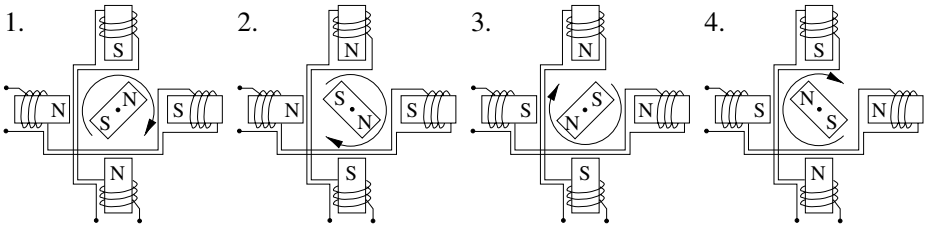


Fig. 1. Simple Stepper Motor Stepping

Changes to coil voltages occur on such a small time scale that their continuous simulation is judged unnecessary for modeling dynamics relevant to the verification task. Voltage changes are therefore approximated as discrete events. Our acceleration control is open-loop: At fixed intervals the motor is stepped according to an acceleration table. We can express such a system as a nonlinear hybrid automaton as shown in Figure 2.

First, we note that there is no apparent “geometrically linear hybrid system”¹ approximation with which we could apply the tools of computational geometry,

¹ i.e. restricted to constant first derivatives; “geometrically” as opposed to “algebraically”

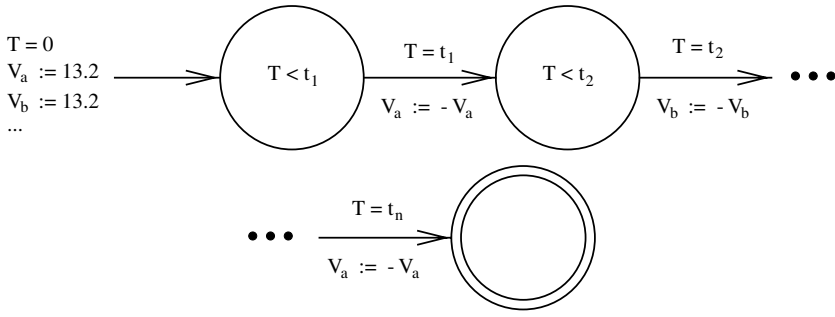


Fig. 2. Stepper Motor Nonlinear Hybrid Automaton

but simulation is feasible. Next, we note that our verification is concerned with a fixed initial time interval (i.e. during acceleration) and is therefore an initial safety problem. Finally, we note that we can compute minimum angular displacement from a stall state over all simulation states as a simple heuristic to numerically rate the relative safety of safe trajectories. We can now ask, “For all possible system parameters and initial states, are all simulation trajectories rated safe?” Put another way, “Is the minimum heuristic evaluation of all possible simulations greater than zero?” If we can answer this optimization question positively, we have verified safety of our hybrid system.

One could argue that such optimization is *not* verification, that one cannot exhaustively simulate all possibilities and can therefore have no guarantees. One can only use such optimization for refutation. To this, we offer two responses: First, if one has additional knowledge of characteristics of one’s heuristic evaluation function (e.g. Lipschitz conditions), then an intelligent optimization approach can utilize such characteristics to guarantee a strictly positive minimum with enough testing. The key is to provide a heuristic evaluation that induces a helpful search landscape without itself become overly burdensome computationally. Second, if one has no such knowledge about the heuristic, the absence of verification techniques well-suited to non-trivial dynamics leaves good global optimization as our best assurance. Our desire is to develop an information-based GO method which, when halted without finding an unsafe trajectory, provides some measure of the thoroughness of its search.

This said, we have endeavored to study a number of representative global optimization techniques in order to assess their suitability to our purpose and point the way towards future innovation.

3 Algorithms and test problems

In this section, we describe the global optimization (GO) algorithms used in this study, the local optimization procedures used by them, and the test functions to be minimized. Author-supplied default settings were used for GO algorithms when possible. Otherwise, reasonable parameters were held constant throughout

testing. Since our goal is to perform a computationally expensive optimization, we would desire an algorithm which reliably and efficiently gives the desired result without tuning. Experienced users of such algorithms applying problem- and domain-specific knowledge to the choice of options and parameters could expect to yield better results.

The first set of algorithms we consider are variants of simulated annealing (SA) [2,3]. SA algorithms are theoretically guaranteed to find the global minimum of a function provided that the annealing schedule starts with sufficiently high temperature and cools sufficiently slowly. However, this guarantee comes at great expense in terms of function evaluations. Finding a suitable annealing schedule which balances the tradeoff of reliability versus efficiency is key to the practicality of SA for our purposes.

AMEBSA [4, pp.451-455] performs SA by modifying a downhill simplex method [4, pp.408-412] such that actual function values of simplex points and possible replacement points are perturbed according to the temperature parameter when making move decisions. Since AMEBSA has no default annealing schedule, we have chosen to use the one supplied in the authors' example [5, pp.182-184]. ASA² [6], "adaptive simulated annealing", is a SA variant that relies on randomly importance-sampling the search space and adapts separate annealing schedules for each parameter. The automatic adaptation of the annealing schedule trades off reliability for efficiency. SALO [7] seeks to combine the theoretical guarantees of SA with the efficiency of local optimization (LO). SALO on f is SA on f' , where f' is f transformed by LO. At each point that SA evaluates, LO takes place and the value of the local minimum is returned. This is intended to "flatten" f and speed convergence to the global minimum. In both implementations described here and in [7], ASA is used as the SA method. In doing so, we again tradeoff reliability for efficiency. When each of these SA methods halts unsuccessfully, it is restarted from the lowest point found thus far.

The second set of algorithms we consider are variants of Multi Level Single Linkage (MLSL) [8]. MLSL uniformly, iteratively samples the search space and performs LO selectively. For each iteration, a new batch of points is evaluated. For each point sampled, LO takes place if there exists no lower sampled point within a critical distance.³ MLSL1 is the original algorithm [8]. MLSLD is a variant of MLSL1 which assumes that the LO procedure is deterministic and should therefore never be repeated from the same sampled point. MLSLO orders optimizations for each iteration by ascending function value of sampled points. MLSLOD has both variations. MLSLSA alternates iterations of MLSLOD with runs of ASA, using the current minimum as the initial point for ASA. LMLSL is a variant of MLSL1 which performs "lazy" function evaluation. That is, the function value of a point is only evaluated when it becomes necessary. This avoids the relatively large initial cost when optimizing simple functions. LMLSL_c is LMLSL using

² ASA software developed by Lester Ingber and other contributors is available at URL <http://www.ingber.com/> or <ftp://ftp.ingber.com>.

³ We used the critical distance parameter $\zeta = 2$ with 100 points generated per iteration.

an ϵ -descent LO procedure. An ϵ -descent procedure guarantees that, for a step greater than ϵ , the function values at ϵ -intervals along the step are sequentially descending.

RANDLO simply performs random local optimizations and is intended to provide a baseline for understanding how well LO knowledge is used by **SALO** and **MLSL** methods. **MONTE** is a Monte Carlo method.

We next describe the local optimization procedures used by some of these global optimization algorithms. **FMINU** and **CONSTR** are **MATLAB**TM optimization functions [9]. **FMINU** performs unconstrained optimization using a quasi-Newton method with a BFGS formula for updating the Hessian matrix approximation. **FMINU _{ϵ}** is an ϵ -descent modification of **FMINU**. **CONSTR** performs constrained optimization using a sequential quadratic programming method. We supply search space bounds and no additional constraints. **YURETMIN** is a variant of Yuret's Masters thesis Procedure 4-1 [10, p.33] which allows specification of search space bounds.

Finally, we reference the objective functions used for comparing the global optimization algorithms. The first part of our study uses functions selected from GO literature and algorithm demonstrations in order to reveal their relative merits. **RAST** is a scaled Rastrigin function [7]. **HUMP** is the six-hump camel-back function [11]. **G-P** is the Goldstein-Price function [11]. **GW1** and **GW100** are 6-dimensional Griewank functions with bounds of each dimension $[-1, 1]$ and $[-100, 100]$ respectively [7]. **SWISS** is a 4-D paraboloid with a lattice of many circular pits [5]. **CMMR** is a 4-D paraboloid with a grid of deep troughs [12]. **GW100**, **SWISS**, and **CMMR** have many local minima. **RAST** has a moderate number. **HUMP**, **G-P**, and **GW1** have few. **RAST**, **GW100**, **SWISS**, and **CMMR** are generally paraboloid in shape with different local minima "traps". All slope up to the bounds of the search space.

The second part of our study concerns the motivating example for this research. Test function **STEP1** takes as input two parameters (viscous friction and load inertia) of the stepper motor model⁴, simulates acceleration of the motor, and performs a simple heuristic evaluation of the trajectory by computing the minimum distance to a stall state (0 if stalled). Such a heuristic function is often simple to construct. **STEP2** is **STEP1** logarithmically scaled so as to focus on the unsafe region of the parameter space. These functions are shown in Figures 3 and 4.

4 Results

Our first tests made use of LO procedure **FMINU** where applicable. 100 optimization trials were performed for each objective function with a maximum of 10000 function evaluations permitted per trial. Each objective function was offset (if necessary) to have a global minimum value of 0. A successful trial was one

⁴ All other parameters and initial conditions are fixed. Given that the original problem posed was too easy, the search space was restricted to provide greater challenge for the methods under comparison.

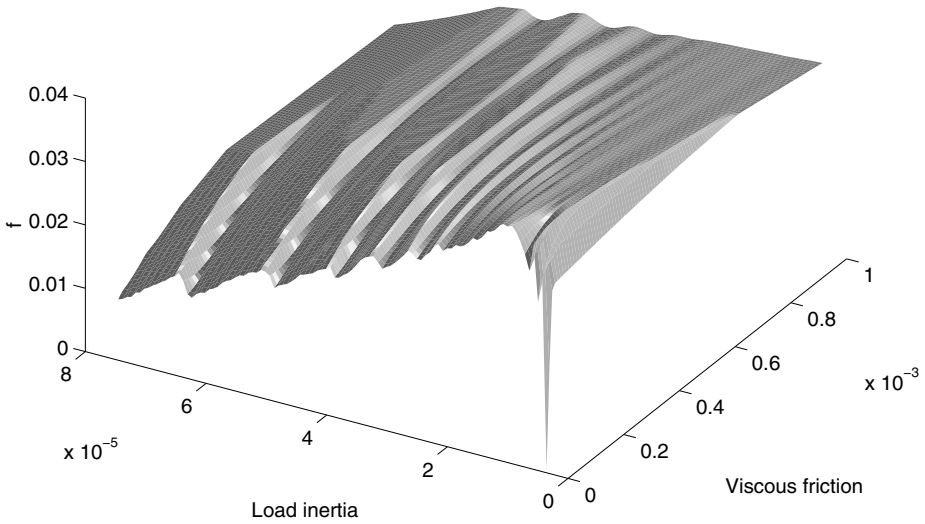


Fig. 3. Stepper Motor Test Function STEP1

in which the optimization routine found a point with function value less than .001 within 10000 function evaluations. This was intended to simulate conditions where the search space contains very small regions with unsafe trajectories.

Each entry in the table of results (Figure 5) shows the number of successful trials (upper left) and the average number of function evaluations for such trials (lower right).

Giving the best performance in nearly half of the tests, **RANDLO** performed surprisingly well, especially for **SWISS** which has a 4-D lattice of numerous “traps”. As **RANDLO**’s LO procedure, **FMINU** is clearly rarely caught in such traps. Since both trap and non-trap regions are paraboloid surfaces, they effectively “point” to the global minimum for LO procedures such as **FMINU**. The simple but important observation here is that local optimization does not necessarily find the *nearest* local optimum. We next observe that both **SALO** and **MLSL** each rely somewhat on *nearness* of LO, and will later turn our attention to the relationship between the global and local layers of each. **FMINU**, which assumes f is continuous, behaved understandably poorly for highly discontinuous **CMMR**. Thus all methods dependent entirely on LO failed all **CMMR** trials. Given that the characteristics of f may not be well understood, this means that a less efficient LO procedure making fewer assumptions would likely be better suited to our purposes.

SALO yielded performance similar to that of **RANDLO** where few LOs sufficed and significantly better where more local optima trapped LO (e.g. in **RAST** and

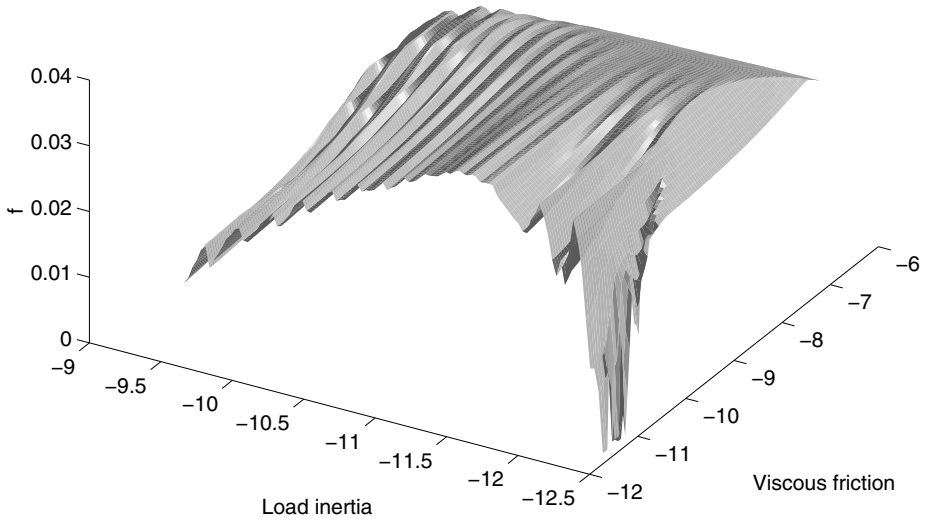


Fig. 4. Stepper Motor Test Function STEP2

GW100). At the heart of **SALO**'s design is the following intention: “SA helps in locating good regions of the search space, while the local optimizer is used to rapidly hit the optimum.” [7] It is clear from this comparison that **SALO** does indeed successfully apply SA on f' to find good regions of f . When comparing **ASA** with **SALO**, it also appears that the cost for transforming $f \xrightarrow{\text{FMINU}} f'$ is usually more than compensated by the efficiency gained.

SALO was designed with hope that f' would be a “simpler” surface than f , reflecting the function value of the nearest optimum. Interestingly, the designers' experiments utilized Yuret's LO procedure which has short term memory and takes increasingly greater steps downhill as success allows. Such a LO procedure can possibly pass over nearest local minima as step size becomes large. Also Yuret's procedure, being stochastic, doesn't simply transform one surface to another. Nevertheless, their experiments and ours indicate that **ASA** is able to handle such LO output gracefully in the long run. The fact that **SALO** outperforms **RANDLO** for harder optimization problems is specifically a property of SA and more generally a form of learning. One can view the changing state probability distribution of SA as a gradual accumulation of knowledge about the location of the global minimum. While such learning is effective given a suitable annealing schedule, it is also weak. Heavily traversed local minima may be heavily traversed

	RAST	HUMP	G-P	GW1	GW100	SWISS	CMMR
AMEBSA	16	100	90	100	0	100	2
	39	40	222	86	N/A	1340	5674
ASA	100	100	100	100	2	100	100
	404	225	1042	197	6003	903	3756
SALO	100	100	100	100	95	100	0
	585	65	97	85	4501	163	N/A
MLSL1	100	100	100	100	47	100	0
	872	154	170	185	4315	239	N/A
MLSLD	100	100	100	100	60	100	0
	636	154	170	185	4492	238	N/A
MLSLOD	100	100	100	100	52	100	0
	556	130	132	173	4370	253	N/A
MLSLSA	100	100	100	100	22	100	99
	544	131	130	174	2609	254	5019
LMLSL	100	100	100	100	50	100	0
	847	105	118	96	4508	187	N/A
LMLSL _ε	100	100	100	100	53	100	0
	638	96	109	93	3864	192	N/A
RANDLO	100	100	100	100	58	100	0
	706	70	96	85	4008	146	N/A

Fig. 5. Successful global optimization trials and average function evaluations

again. All but one of the function evaluations made in LO are ignored. Much information is wasted. Nonetheless, SALO's performance was impressive.

Performance of MLSL methods, though similar to that of RANDLO, yields little to commend them over RANDLO. That selective uniform random LO should perform worse than unselective uniform random LO suggests an assumption in MLSL which is not met in our study. Following the analysis more closely in both [13] and [8], we see that MLSL's LO procedure is assumed to be an ϵ -descent procedure such that the current critical distance effectively bounds the step size of LO.⁵ We therefore created FMINU_ε, an ϵ -descent modification of FMINU, and tested LMLSL_ε for comparison. Although LMLSL_ε is somewhat of an improvement over LMLSL, it is still generally worse than RANDLO. ϵ -descent does not therefore appear to help us much. We conjecture that MLSL methods dominate RANDLO for objective functions where LO is trapped in many minima, and that SALO dominates MLSL methods for such objective functions in our study because our f' -surfaces are easily globally optimized with LO. To elucidate the latter point, consider RAST, GW100, and SWISS. LO roughly transforms each into a paraboloid of plateaus. LO of such LO-transformed functions can then efficiently lead to the global optimum. We can view the task of global optimization as *multi-level local optimization*. The base-level LO₀ takes advantage of whatever information about f is available (continuity, gradients, etc.), the next level LO₁ is suited to the class

⁵ This is nowhere mentioned in survey [14] and is not emphasized elsewhere in the literature.

of one’s LO_0 -transformed function f' , and so on. We may stop after arbitrarily many (probably 2-3) LO levels and simply perform random local optimization. The role of each LO level is to enlarge the regions leading to global optima. We are currently developing such multi-level local optimization methods.

Regarding MLSL methods, let us also note that, like SALO, they all but ignore information gained through LO. Uniformly sampled points are locally optimized based only on the values of sampled points within a critical distance. Again we find great waste of information gained at great expense.

AMEBSA gave mixed results which can likely be attributed to the lack of annealing schedule tuning. Perhaps an adaptive annealing schedule would make AMEBSA more suitable for such problems. ASA’s efficiency was unpredictable, although it was perhaps the most reliable method for this set of objective functions.

While these functions may give a general indication of the relative strengths of these methods without tuning, the functions share a common property undesirable for our purposes: The *unconstrained* global minimum is never located at or beyond the bounds of the search space. Therefore, our optimization methods need not perform well along the bounds of our search space. It is for this reason that unconstrained FMINU was suitable for use with such global optimizations. We used this as an opportunity to try two constrained LO procedures CONSTR and YURETMIN for the stepper motor test problems STEP1 and STEP2. For this testing, we performed 10 trials to find a function value of 0 with a maximum of 1000 function evaluations per trial. The results appear in the tables of figure 6.

	STEP1	STEP2		STEP1	STEP2
ASA	0	2	ASA	0	2
	N/A	497		N/A	497
SALO	10	5	SALO	7	9
	80	202		387	198
MLSLOD	10	10	MLSLO	4	10
	127	191		790	231
LMLSL	10	10	LMLSL	3	10
	163	137		389	169
RANDLO	10	10	RANDLO	9	10
	78	359		501	172
MONTE	0	6	MONTE	0	6
	N/A	469		N/A	469

(a) CONSTR

(b) YURETMIN

Fig. 6. Results for STEP1 and STEP2

Since both STEP1 and STEP2 have a small number of local minima along the bounds of the search space, behavior of LO again figured most significantly in our results. Despite the fact that much of the search space slopes downward away

from the corner where failures occur, **CONSTR** had a bias towards looking in that particular corner. It was thought that **STEP2** (log-log scaled **STEP1**) would be an easier function to optimize, but this was not the case. Not only was the global minimum basin expanded, but nearby local minima also expanded, trapping **LO** more often.

ASA's function evaluation expenses were such that it was outperformed by **MONTE**. The remaining **LO**-based methods performed similarly overall. The cost of computing simple heuristic information about relative safety of trajectories is usually more than compensated by efficiency in discovering unsafe trajectories through optimization. For both **LO** procedures, **RANDLO** gave best performance for **STEP1**, and **LMSL** gave best performance for **STEP2**. Although there was no universal "winner" among global optimization procedures, it is encouraging to note that procedures such as **SALO** and **LMSL** could be run in parallel to achieve respectable, more reliable results. The choice of **LO** procedure proved very significant for performance, which again underscores the importance of developing robust, efficient **LO** procedures suited to large classes of functions.

5 Conclusions

A powerful approach to initial safety verification is to transform the problem into an optimization problem and leverage the power of efficient optimization methods.

This is accomplished by

- providing a good heuristic evaluation function f ,
- choosing an efficient local optimization procedure well suited to f , and
- applying a global optimization procedure such as **SALO** with **ASA** or **RANDLO** for which one's local optimization procedure is well suited.

For our stepper motor problem, we are fortunate to have system trajectories which are continuous in the initial condition. Our heuristic function preserves such continuity, so we can use such knowledge when choosing our local optimization procedure. However, if one cannot assume a continuous heuristic function (as will often be the case with hybrid systems), one can use the same approach choosing global and local optimization methods according to one's assumptions or lack thereof.

While no global optimization procedure was generally dominant in our comparative study, random local optimization seemed best suited for objective functions with few minima, and **SALO** with **ASA** seemed best suited for objective functions with many minima. By making use of **ASA** for **SA**, one both avoids the need to specify an annealing schedule and benefits from its relative efficiency among **SA** algorithms. Although one is encouraged to make use of **ASA**'s options to improve performance, we have not done so and have been pleased with most results nevertheless.

SALO and MLSL methods perform global optimization with global and local search phases, and rely on local optimization for efficiency. However, both methods make little or no use of information gained in the course of local optimization. We believe that great progress will be made in global optimization when global optimization and local optimization are seamlessly integrated to share knowledge gained of f . Where evaluation of f is computationally expensive, it is worth computational expense to utilize such knowledge for the efficiency of global optimization. To this end, we are currently developing a set of information-based optimization techniques where each optimization step is chosen with respect to the information gained thus far.

References

1. Albert C. Leenhouts. *Step Motor System Design Handbook*. Litchfield Engineering, Kingman, Arizona, USA, 1991.
2. N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *J. Chem. Phys.*, 21(6):1087–1092, 1953.
3. S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.
4. William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: the art of scientific computing - 2nd Ed.* Cambridge University Press, Cambridge, 1992.
5. William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes Example Book (C) - 2nd Ed.* Cambridge University Press, Cambridge, 1992.
6. Lester Ingber. Adaptive simulated annealing (ASA): Lessons learned. *Control and Cybernetics*, 25(1):33–54, 1996.
7. Rutvik Desai and Rajendra Patil. SALO: combining simulated annealing and local optimization for efficient global optimization. In J.H. Stewman, editor, *Proceedings of the 9th Florida AI Research Symposium (FLAIRS-'96)*, pages 233–237, St. Petersburg, FL, USA, 1996. Eckerd Coll.
8. A.H.G. Rinnooy Kan and G.T. Timmer. Stochastic global optimization methods; part II: clustering methods. *Mathematical Programming*, 39:57–78, 1987.
9. Andrew Grace. *Optimization Toolbox*. The Mathworks Inc., 24 Prime Park Way, Natick, MA 01760-1500 USA.
10. Dinez Yuret. From genetic algorithms to efficient optimization. Master's thesis, Massachusetts Institute of Technology, May 1994.
11. F.H. Branin. Widely convergent method for finding multiple solutions of simultaneous nonlinear equations. *I.B.M. J. R&D.*, Sept 1972.
12. A. Corana, M. Marchesi, C. Martini, and S. Ridella. Minimizing multimodal functions of continuous variables with the “simulated annealing” algorithm. *ACM Trans. Mathl. Software*, 13(3):262–279, 1987.
13. A.H.G. Rinnooy Kan and G.T. Timmer. Stochastic global optimization methods; part I: multi level methods. *Mathematical Programming*, 39:27–56, 1987.
14. C. Guus E. Boender and H. Edwin Romeijn. Stochastic methods. In Horst and Pardalos [15], pages 829–869.
15. Reiner Horst and Ralos M. Pardalos, editors. *Handbook of Global Optimization*. Kluwer Academic, Dordrecht, Netherlands, 1995.

A Hybrid Receding–Horizon Control Scheme for Nonlinear Discrete–Time Systems

T. Parisini¹ and S. Sacone^{2*}

¹ Department of Electrical, Electronic and Computer Engineering,
DEEI–University of Trieste, Via Valerio 10, 34175 Trieste, Italy
`parisini@univ.trieste.it`

² Department of Communications, Computer and System Sciences
DIST–University of Genoa, Via Opera Pia 13, 16145 Genova, Italy
`simona@dist.unige.it`

Abstract. Hybrid systems made up of a continuous plant supervised by a discrete–event controller are considered in this paper. A set of optimal feedback control functions is defined at the continuous level. The aim of the discrete–event controller, modelled by means of a finite-state automaton, is that of choosing, in the above mentioned set, the best control action to be applied to the plant in dependence of the current plant conditions and of possibly occurred external events (hence, the overall control scheme can be viewed as a two-level hybrid control scheme). An invariance property of the hybrid control scheme is proved and extensive simulation results are reported, showing the effectiveness of the proposed methodology.

1 Introduction

The contemporary presence of several kinds of dynamic behaviours within the same dynamical system represents the major feature of the so-called *hybrid dynamical systems*. An hybrid system is typically made up of a continuous plant whose behaviour is logically controlled by a discrete–event supervisor. Many real systems are intrinsically structured as hybrid systems; examples include manufacturing systems, computer communication networks, constrained robotic systems, process control systems, and transportation systems.

In the last years, the attention of many researchers has, thus, been focused on the possibility of modelling, analysing, and controlling the behaviour of hybrid systems. Modelling frameworks of hybrid systems have been proposed, for instance, in [1], [2], and [3], whereas in [4], the authors suggest a general hybrid model also providing a survey of related results. As regards hybrid control schemes, many different approaches can be found in the literature. In [5] and [6],

* This work was partly supported by the Italian Ministry for the University and Research (MURST), by the Italian Research Council(CNR), and by the Italian Space Agency (ASI).

some general concepts of the supervisory control theory suggested by Ramadge-Wonham are applied to hybrid systems. A sliding-mode control approach specifically developed for hybrid systems is proposed in [7], whereas a multilayer hierarchical control problem is applied in [8] to Intelligent Vehicle/Highway Systems, a particular example of hybrid systems in the field of transportation. A further hybrid control scheme is, then, formalized in [9] by means of H^∞ control concepts.

A special class of hybrid systems is represented by *switched systems*, defined as combinations of finitely many continuous dynamical systems. A first qualitative analysis of the behaviour of switched systems is addressed in [10] in which "multiple Lyapunov functions" are used. An extensive study of stability criteria for switched systems is reported in [11] where the authors establish new results regarding the stability and asymptotic stability of the equilibrium of such systems. These results are mainly relevant to the construction of Lyapunov functions which, as already done by the same authors for discontinuous dynamical systems [12], allow the establishment of sufficient conditions for stability and asymptotic stability of the equilibrium of the considered systems.

The model adopted in this work is based on the modelling concepts proposed in [13] and [14], whereas the hybrid control scheme can be viewed as the juxtaposition of two control levels: a continuous-state feedback controller and a discrete-event supervisor. At the continuous-state level a set of optimal receding-horizon feedback control functions are defined; the discrete-event supervisor acts by choosing the best continuous-state controller depending on current informations about the plant and about external conditions. The invariance properties of the control scheme formalized in this paper are investigated, and a simple analytical result is proved.

The remainder of the paper is organized as follows. In Section 2, the hybrid control scheme is preliminarily described. A detailed description of the feedback control scheme in the continuous-state level is reported in Section 3, whereas the discrete-event supervisor and the interface between the two control levels are outlined in the subsequent sections. In Section 6, the above invariance property is stated and proved, whereas Section 7 presents extensive simulation results on a significant nonlinear control problem, showing the effectiveness of the proposed approach.

2 Definition of the hybrid control scheme

Let us consider the discrete-time dynamic system (in general, nonlinear)

$$x_{t+1} = f(x_t, u_t), \quad t = 0, 1, \dots \quad (1)$$

where $x_t \in X \subset \mathbb{R}^n$ and $u_t \in U \subset \mathbb{R}^m$ are the state and control vectors, respectively. We assume that the constraint regions X and U belong to the class $\mathbb{Z}$ of compact sets containing the origin as an *internal point*. Moreover, we assume that $f \in \mathcal{C}^1 [\mathbb{R}^n \times \mathbb{R}^m, \mathbb{R}^n]$, with $f(0, 0) = 0$.

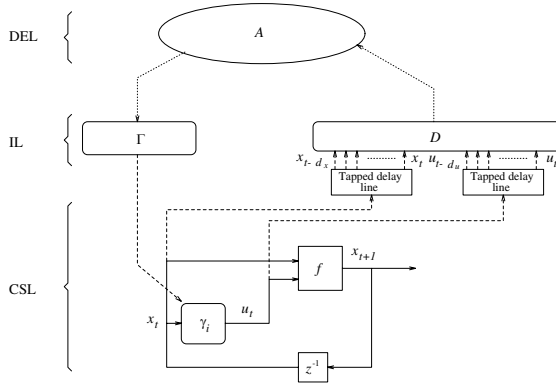


Fig. 1. Structure of the hybrid control scheme.

The structure of the hybrid-control scheme is shown in Fig. 1. This scheme has some similarities and analogies with the one reported in [15].

As shown in Fig. 1, three different levels are present: a *continuous-state level* (CSL), and *interface level* (IL) and a *discrete-event level* (DES). In the following three sections, a precise definition of these levels will be given. However, let us first describe the rationale for choosing such an hybrid scheme. In this respect, we would like to emphasize that the approach pursued in the present work is very preliminar and, as already mentioned in the Introduction, many more general schemes of hybrid controllers can be found in the literature.

In very rough terms (as just mentioned above precise statements will be given in Sections 3, 4 and 5), the hybrid-control scheme behaves as follows: a set Γ of feedback control functions $\gamma_i(\cdot)$ is given; each instance of the closed-loop system $[f, \gamma_i]$ in the CSL is characterized by some typical behavior $\mathcal{B}_i$. In the IL, at each time instant t , some decision algorithm $\mathcal{D}$ elaborates batches of state vectors $\{x_{t-d_x}, \dots, x_t\}$ and control vectors $\{u_{t-d_u}, \dots, u_t\}$ thus generating some aggregate information vector $\mathcal{I}_t$. The discrete-event system A in the DEL reacts to the input vector $\mathcal{I}_t$ and to possibly occurred external events by maintaining or changing the control function γ_i .

3 The feedback control scheme in the CSL

In the present section, a precise characterization of the nonlinear feedback control scheme is given. To this end, let us define the class of control functions Γ as

$$\Gamma \triangleq \{ \gamma_i : \mathcal{X}_i \longrightarrow \mathbb{R}^m, i \in I \} \quad (2)$$

where $\mathcal{X}_i \subseteq X$, $\mathcal{X}_i \in \mathbb{Z}$, $\forall i \in I$ and where I is a set of integers with $\text{card}(I) = \psi$ allowing the choice of a single $\gamma_i \in \Gamma$. Clearly, this definition of Γ is too abstract. Then, a more specific set of control functions is addressed. We consider the class of discrete-time *receding-horizon* (RH) regulators (see, for instance, [16],[17]).

To this end, for any $i \in I$, we need to define the following *finite-horizon* cost function

$$J_{FH}^i(x_t, u_{t,t+N-1}, N^i, a^i, P^i) = \sum_{j=t}^{t+N^i-1} h(x_j, u_j) + a^i \|x_{t+N}\|_{P^i}^2, \quad t \geq 0 \quad (3)$$

where $u_{t\tau} \triangleq \text{col}(u_t, \dots, u_\tau)$, $h \in \mathcal{C}^1[\mathbb{R}^n \times \mathbb{R}^m, \mathbb{R}^+]$, with $h(0,0) = 0$., $\|x\|_{P^i}^2 \triangleq x^T P^i x$, $a^i \in \mathbb{R}$ is a positive scalar, $P^i \in \mathbb{R}^{n \times n}$ is a positive-definite symmetric matrix, and N^i is a positive integer denoting the length of the control horizon. Then we can state the following

Problem 1. For any $i \in I$ and at every time instant $t \geq 0$, find the RH optimal control law $u_t^{RH^\circ} = \gamma_{RH}^{i^\circ}(x_t) \in U$, where $u_t^{RH^\circ}$ is the first vector of the control sequence, $u_t^{FH^\circ}, \dots, u_{t+N-1}^{FH^\circ}$ (i.e., $u_t^{RH^\circ} \triangleq u_t^{FH^\circ}$), that minimizes the cost (3) for the state $x_t \in X$.

Remark 1. As will be clear in the following, for any given $i \in I$, the final cost $a^i \|x_{t+N}\|_{P^i}^2$ plays a key role in the development of stability results concerning the RH regulator, and is essentially a penalty function related to the relaxation of the terminal state constraint $x_{t+N} = 0$, imposed in [18] and in [16]. In the following, a basic result will be reported, showing the effects of the choice of N^i , a^i , and P^i on the stability and performances of the i -th RH regulator. As will be seen later on, this result will be crucial in order to define in an effective way the interaction of the three levels in the hybrid scheme.

Remark 2. The statement of Problem 2 does not impose any particular way of computing the control vector $u_t^{RH^\circ}$ as a function of x_t . Actually, we have two possibilities: i) *on-line computation*. For any $i \in I$, Problem 1 is an open-loop optimal control problem and may be regarded as a nonlinear programming one. The main advantage of this approach (adopted in the works by Keerthi and Gilbert and by Mayne and Michalska) is that many well-established nonlinear programming techniques are available to solve Problem 1. ii) *Off-line computation*. This approach implies that the control laws $\gamma_{RH}^{i^\circ}(x_t)$, $i \in I$ have to be computed “a priori” and stored in the regulator’s memory. Clearly, the off-line computation has advantages and disadvantages that are opposite to the ones of the on-line approach. No on-line computational effort is requested from the regulator, but a very large amount of computer memory may be required to store the closed-loop control law.

Now we state the basic stability result for the above RH controller (more details can be found in [19]). In this connection the following assumptions are needed:

- (i) The linear system $x_{t+1} = Ax_t + Bu_t$, obtained via the linearization of the system (1) in a neighborhood of the origin (i.e., $A \triangleq \frac{\partial f}{\partial x_t} \Big|_{x_t=0, u_t=0}$ and $B \triangleq \frac{\partial f}{\partial u_t} \Big|_{x_t=0, u_t=0}$), is stabilizable.
- (ii) The transition cost function $h(x, u)$ depends on both x and u , and there exist two strictly increasing functions $r, s \in \mathcal{C}[\mathbb{R}^+, \mathbb{R}^+]$, with $r(0) = s(0) = 0$, such that $r(\|(x, u)\|) \leq h(x, u) \leq s(\|(x, u)\|)$, $\forall x \in X, \forall u \in U$, where $(x, u) \triangleq \text{col}(x, u)$.
- (iii) There exists a compact set $X_0 \subseteq X, X_0 \in \mathbb{Z}$, with the property that there exists a control horizon $M \geq 1$ such that there exists a sequence of admissible control vectors $\{u_i \in U, i = t, \dots, t + M - 1\}$ that yield an admissible state trajectory $x_i \in X, i = t, t + 1, \dots, t + M$ ending in the origin of the state space (i.e., $x_{t+M} = 0$) for any initial state $x_t \in X_0$.
- (iv) The optimal finite-horizon feedback control functions $\gamma_{FH}^{\circ}(x_j, j), j = t, \dots, t + N^i - 1$, which minimize the i -th cost (3), are continuous with respect to x_j , for any $x_t \in X$ and any finite integer $N^i \geq 1$.

Note that assumption (iii) substantially concerns the controllability of the nonlinear system (1). Let us now denote by $J_{FH}^{\circ}(x_t, N^i, a^i, P^i) \triangleq J_{FH}^i(x_t, u_{t,t+N-1}^{\circ}, N^i, a^i, P^i) = \sum_{j=t}^{t+N-1} h(x_j^{FH^{\circ}}, u_j^{FH^{\circ}}) + h_F^i(x_{t+N}^{FH^{\circ}})$ the cost corresponding to the i -th optimal N -stage trajectory starting from x_t (to simplify the notation, we let $h_F^i(x) \triangleq a^i \|x\|_{P^i}^2$, without any ambiguity whenever a^i and P^i need not be rendered explicit). Then the following theorem can be proved.

Theorem 1.[19] *Let us consider a generic $i \in I$. If assumptions (i) to (iv) are verified, there exist a positive scalar $\tilde{a}^i$ and a positive-definite symmetric matrix $P^i \in \mathbb{R}^{n \times n}$ such that, for any $N^i \geq M$ and any $a^i \in \mathbb{R}, a^i \geq \tilde{a}^i$, the following properties hold:*

- 1) *The RH control law stabilizes asymptotically the origin, which is an equilibrium point of the resulting closed-loop system.*
- 2) *There exists a positive scalar β^i such that the set $\mathcal{W}(N^i, a^i, P^i) \in \mathbb{Z}$, $\mathcal{W}(N^i, a^i, P^i) \triangleq \{x \in X : J_{FH}^{\circ}(x, N^i, a^i, P^i) \leq \beta^i\}$, is an invariant subset of X_0 and a domain of attraction for the origin, i.e., for any $x_t \in \mathcal{W}(N^i, a^i, P^i)$, the state trajectory generated by the RH regulator remains entirely contained in $\mathcal{W}(N^i, a^i, P^i)$ and converges to the origin.*

Remark 3. More details on the above theorem can be found in [19]. We want only to remark that Theorem 1 is constructive in the sense that, for any N^i , the scalar $\tilde{a}^i$ and the matrix P^i can be determined explicitly, thus making the design of the RH regulator implementable.

4 The interface level

As already mentioned in the Introduction, the discrete-event level and the continuous-state level communicate by means of an interface level made up of two distinct parts: a first module implementing the algorithm $\mathcal{D}$ which transforms the batches of the state and control vectors of the plant into an aggregate information vector $\mathcal{I}_t^f$, and a second module simply composed of the set Γ of possible control functions.

In general, the availability of the batches $\{x_{t-d_x}, \dots, x_t\}$ and control vectors $\{u_{t-d_u}, \dots, u_t\}$ allows to perform very general operations like, for instance, the identification of a model of the plant, the condition monitoring of the closed-loop system, and so on. In the present preliminary work we consider the very simple situation $d_x = d_u = 0$, but this is not restrictive.

We now detail a little more the algorithm $\mathcal{D}$. At each time instant t it receives as an input the values x_t and u_t coming from the CSL. On the basis of these values, $\mathcal{D}$ defines the set $\mathcal{I}(x_t) \triangleq \{i \in I : x_t \in \mathcal{X}_i\}$. Of course, it is necessary to use here a set of regions since x_t could belong to the intersection of two or more regions. Moreover, $\mathcal{D}$ identifies the index i_t of the control action presently applied and the set of feasible control functions for time instant $t + 1$. Then, in addition to $\mathcal{I}(x_t)$, we introduce also a set of feasible control functions $\mathcal{J}(x_t)$. As will be seen in Section 6, this set is built taking into account some stability criteria. Clearly, in a more general setting, this feasible set could be characterized in a different way. The aggregate information vector $\mathcal{I}_t^f$ is now defined as

$$\mathcal{I}_t^f(x_t) \triangleq \text{col}[\mathcal{I}(x_t), i_t, \mathcal{J}(x_t)]$$

The second module of the IL is very simple since it receives from the DEL the index $i, i \in I$, of the control strategy to be applied in the CSL and it sets up the CSL regulator with γ_i .

5 Discrete-event system model

The DEL, which constitutes the discrete-event supervisor, is modelled as a deterministic automaton defined as the seven-tuple $(\tilde{E}, \tilde{X}, \Sigma, \mu, \tilde{\delta}, \phi, \Lambda)$, where $\tilde{E}$ is the automaton *alphabet*, that is a finite set of symbols in which each symbol identifies a specific event, $\tilde{X}$ is the finite *state set* of the automaton, $\Sigma(\xi)$ is the *feasible (or enabled) event set*, defined for all $\xi \in \tilde{X}$ with $\Sigma(\xi) \subseteq \tilde{E}$, μ_t is the automaton *input function*, $\mu_t : I^{\psi_t} \rightarrow \tilde{X}$ ($\psi_t \triangleq \text{card}[\mathcal{I}_t^f(x_t)]$), $\tilde{\delta}$ is the automaton *state transition function*, $\tilde{\delta} : \tilde{X} \times \tilde{E} \rightarrow \tilde{X}$, ϕ is the automaton *output function*, $\phi : \tilde{X} \rightarrow I$, Λ is the automaton *clock structure*. The clock structure Λ is defined as $\Lambda = \{\tau_j, j = 1, 2, \dots\}$ defining τ_j as the time instant of occurrence of the j -th event (of any type).

The DEL receives as input the aggregate information vector $\mathcal{I}_t^f$ bringing information about the CSL as generated by the IL. The automaton input function μ_t transforms the information vector $\mathcal{I}_t^f$ into the corresponding automaton

state $\xi \in \tilde{X}$. Then, the dynamic behaviour of the automaton A obeys to the rule defined by the state transition function, which means that the next automaton state ξ' can be expressed as:

$$\xi' = \tilde{\delta}(\xi, e) \quad (4)$$

where ξ is the automaton current state and $e \in \Sigma(\xi)$ is the next event occurring. To analyze the automaton dynamics in a synchronous way, it is necessary to define the automaton state in relation to a specific discrete-time instant t . To do this, let us define ξ_t as the automaton state at time instant t . Of course, the time instant t is referred to the synchronous clock structure of the CSL and it is not necessarily included in the automaton clock structure A . The structure of the automaton state ξ_t is defined as

$$\xi_t \triangleq \text{col}[\mathcal{I}_t^f(x_t), e_{[t-1, t)}]$$

where $e_{[t-1, t)}$ is the sequence of events occurring in the time-window $[t-1, t)$. Equation 4 becomes now

$$\xi_{t+1} = \delta(\xi_t, e_{[t, t+1)}) \quad (5)$$

where $e_{[t, t+1)}$ denotes the sequence of events occurring in the time-window $[t, t+1)$. More specifically, we have

$$e_{[t, t+1)} \triangleq \begin{cases} e_{\emptyset} & \text{if } \nexists j: \tau_j \in [t, t+1) \\ \{e_s, j^- \leq s \leq j^+\}, \text{ with } j^-, j^+: \tau_{j^-}, \dots, \tau_{j^+} \in [t, t+1) & \text{otherwise} \end{cases}$$

where $e_{\emptyset}$ denotes the null event, which means that no external event has occurred in $[t, t+1)$. The state transition function δ is defined in a straightforward way: by using the compact notation $\tilde{\delta}^e(\xi) \triangleq \tilde{\delta}(\xi, e)$, we have (the symbol $\circ$ denotes the composition)

$$\xi_{t+1} = \delta(\xi_t, e_{[t, t+1)}) \triangleq \tilde{\delta}^{e_{\tau_{j^+}}} \circ \dots \circ \tilde{\delta}^{e_{\tau_{j^-}}}(\xi_t)$$

Remark 4. For most applications, we expect that the rate of event-changes is much slower than the sampling rate. Equivalently, it is very unlikely that, for some t , there exist τ_j and τ_{j+1} such that $t \leq \tau_j < \tau_{j+1} < t+1$.

The DEL objective is that of choosing, depending on the current automaton state and on the sequence of events occurred in the last considered time-window, the control action to be adopted in the next time interval. To do this, the DEL includes a *decisional* look-up table $\mathcal{T}$ in which each combination of a vector $\mathcal{I}_t^f(x)$ and of a type of event is associated with the index of the best control function to use. This means that each entry of $\mathcal{T}$ is constituted by a triple $\{\mathcal{I}_t^f(x), e, i\}$, $x \in X, e \in \tilde{E}, i \in I$. It is important to remark here that the best control strategy to be adopted at time interval $t+1$ can be different from the one applied in t even if no external event occurs in $[t, t+1)$, but depending only on the system state ξ_t .

Thus, in the time interval $(t, t+1)$, the state transition function δ works in this way:

1. on the basis of ξ_t and $e_{[t,t+1]}$ it identifies the index $j, j \in I$ of the best control strategy to be applied;
2. it checks whether $j \in \mathcal{I}_t^f(x_t)$ or not;
3. in the affirmative case, j is inserted in the next automaton state, otherwise the new control strategy is defined as that with index belonging to $\mathcal{I}_t^f(x_t)$ and with minimum distance from γ_j , in some specified metric.

The automaton output function ϕ defines the correspondence between the automaton states and the integer indexes identifying the different control strategies. Thus, ϕ simply realizes the mapping between an automaton state $\xi \in \tilde{X}$ and the index $i \in I$ corresponding to the i -th control strategy γ_i . In this way, the DEL can change the control strategy to be applied to the plant, in dependence of some information about the plant state vector and control vector, and, above all, according to the possible occurrence of asynchronous external events. As an example, a simple re-configuration scheme in a diagnostic system can be easily put in the form just presented.

6 An invariance property of the hybrid control scheme

In the present section, an invariance property of the hybrid control scheme is addressed. More specifically, it will be shown that all trajectories of the hybrid system lie in a suitable invariant set.

Let us recall Theorem 1 in Section 3 and define the sets $\mathcal{X}_i \triangleq \mathcal{W}(N^i, a^i, P^i), i \in I$. Clearly, we have that

$$\mathcal{X}_i \subseteq f[\mathcal{X}_i, \gamma_i(\mathcal{X}_i)], \quad i \in I \quad (6)$$

Notice that the evolution of the hybrid-control scheme can be described by the *coupled equations*

$$\begin{cases} \xi_t = \mu_t[\mathcal{I}_t^f(x_t)] \\ \xi_{t+1} = \delta(\xi_t, e_{[t,t+1]}) \\ x_{t+1} = f[x_t, \gamma_{\phi(\xi_{t+1})}(x_t)] \end{cases}$$

Let us now introduce the mapping

$$\mathcal{F}: X \times \tilde{X} \times \tilde{E}_t \longrightarrow \mathbb{R}^n, \quad \mathcal{F}(x_t, \xi_t, e_{[t,t+1]}) \triangleq f[x_t, \gamma_{\phi(\delta(\xi_t, e_{[t,t+1]})})(x_t)]$$

where

$$\tilde{E}_t \triangleq \begin{cases} \emptyset & \text{if } \nexists j \in \mathbb{Z}^+ : \tau_j \in [t, t+1) \\ \underbrace{\tilde{E} \times \dots \times \tilde{E}}_{j^+ - j^- + 1 \text{ times}} & \text{if } \exists j^-, j^+ \in \mathbb{Z}^+ : \tau_{j^-}, \dots, \tau_{j^+} \in [t, t+1) \end{cases}$$

In Proposition 1 below, the invariance property will be stated in terms of the mapping $\mathcal{F}$. Before stating and proving the invariance result, some further useful quantities and notations should be introduced.

$$d_{ij}(x) \triangleq \|\gamma_i(x) - \gamma_j(x)\|, \quad \forall x \in X, \forall i, j \in I, i \neq j \quad (7)$$

$$\Delta_i(x) \triangleq J_{FH}^{i\circ}(x, N^i, a^i, P^i) - J_{FH}^{i\circ}[f(x, \gamma_i(x)), N^i, a^i, P^i], \quad \forall x \in \mathcal{X}_i, \forall i \in I \quad (8)$$

$$k_{FH}^i \triangleq k \in \mathbb{R}^+ : \left\| J_{FH}^{i\circ}(x', N^i, a^i, P^i) - J_{FH}^{i\circ}(x'', N^i, a^i, P^i) \right\| \leq k \|x' - x''\| \\ \forall x', x'' \in X, \quad \forall i \in I \quad (9)$$

$$k_f(x) \triangleq k \in \mathbb{R}^+ : \|f(x, u') - f(x, u'')\| \leq k \|u' - u''\|, \quad \forall u', u'' \in U \quad (10)$$

Finally, let $\mathcal{J}(x_t) \triangleq \{j \in I : d_{ij}(x_t) < \Delta_i(x_t)/k_{FH}^i k_f(x_t), \forall i \in \mathcal{I}(x_t)\}$, and introduce the set $\mathcal{X}^+ \triangleq \bigcup_{i \in I} \mathcal{X}_i$. Clearly, we now have that $\mathcal{I}(x_t) \subseteq \mathcal{J}(x_t)$. In

this connection, we can write $\mathcal{I}_t^f(x_t) \triangleq \text{col}[\mathcal{I}(x_t), i_t, \mathcal{J}(x_t) \setminus \mathcal{I}(x_t)]$, and we have that $\psi_t = \text{card}(\mathcal{J}(x_t)) + 1$. Then, the following result can be proved.

Proposition 1. *First, assume that Theorem 1 holds true $\forall i \in I$. Moreover, assume that the following constraint is embedded in the DEL: if $x_t \in \mathcal{X}^+$, then $\phi(\delta(\xi_t, e_{[t, t+1)})) \in \mathcal{J}(x_t)$, irrespective of the specific values of the arguments $\xi_t, e_{[t, t+1)}$. Then, the set $\mathcal{X}^+$ is controlled-left-invariant with respect to $\tilde{X} \times \tilde{E}_t$ and the mapping $\mathcal{F}$, that is*

$$\forall t \geq 0, \quad \mathcal{F}[\mathcal{X}^+, \tilde{X}, \tilde{E}_t] \subseteq \mathcal{X}^+ \quad (11)$$

Proof of Proposition 1. Let us consider a generic time-instant $t \geq 0$, and generic $\xi \in \tilde{X}$, $e_{[t, t+1)} \in \tilde{E}_t$. We need to prove that $\mathcal{F}(x_t, \xi_t, e_{[t, t+1)}) \in \mathcal{X}^+$, $\forall x_t \in \mathcal{X}^+$. Two different cases have to be addressed.

1. $x_t \in \mathcal{X}^+$ and $\phi(\delta(\xi_t, e_{[t, t+1)})) = j \in \mathcal{I}(x_t)$. Thanks to the constraint in the statement of the proposition and to the invariance of $\mathcal{X}_j$ (see Theorem 1), we have $\mathcal{F}(x_t, \xi_t, e_{[t, t+1)}) \in \mathcal{X}_j \subseteq \mathcal{X}^+$.
2. $x_t \in \mathcal{X}^+$ and $\phi(\delta(\xi_t, e_{[t, t+1)})) = j \in \mathcal{J}(x_t) \setminus \mathcal{I}(x_t)$. In this case, the proof is a little more involved. For any $i \in \mathcal{I}(x_t)$, the set $\mathcal{X}_i$ is an invariant set under the action of the control function γ_i , and the function $V_i(x_t) \triangleq J_{FH}^{i\circ}(x_t, N^i, a^i, P^i)$ is the corresponding Lyapunov function (see [19]). Hence, we have:

$$V_i(f(x_t, \gamma_i(x_t))) - V_i(x_t) = -\Delta_i(x_t) < 0$$

Now, by using constraint 2, we have

$$\begin{aligned} \left| V_i(x_{t+1}^i) - V_i(x_{t+1}^j) \right| &= |V_i(f(x_t, \gamma_i(x_t))) - V_i(f(x_t, \gamma_j(x_t)))| \\ &\leq k_{FH}^i \|f(x_t, \gamma_i(x_t)) - f(x_t, \gamma_j(x_t))\| \leq k_{FH}^i k_f(x_t) \|\gamma_i(x_t) - \gamma_j(x_t)\| \\ &\leq k_{FH}^i k_f(x_t) d_{ij}(x_t) < -\Delta_i(x_t) < 0 \end{aligned}$$

Then V_i is a Lyapunov function and $\mathcal{X}_i \subseteq \mathcal{X}^+$ is an invariant set under the action of γ_j , thus proving that $\mathcal{F}(x_t, \xi_t, e_{[t, t+1)}) \subseteq \mathcal{X}^+$. ■

Remark 5. It is worth noting the meaning of the constraint imposed in the DEL (see Proposition 1). Actually, the set of indexes $\mathcal{J}(x_t)$ denotes the control functions that, in connection with the present state x_t , do not give rise to control actions with too large deviations with respect to those a-priori guaranteeing the decrease of some Lyapunov function (see also Theorem 1).

7 An application of the hybrid controller

In the present section, a complete simulation example for the proposed hybrid receding-horizon controller is presented. First, the description of the system dynamics and of the proposed control objective is given. Then, the definition of the state automaton realizing the DEL and of the corresponding transition function is reported. Finally, significant simulation results are presented and discussed with particular emphasis to the comparison between the application of the proposed hybrid control scheme and that of a control scheme including only the CSL.

7.1 The considered nonlinear system to be controlled

The system considered in this example is relevant to a high-performance aircraft affected by wing-rock when flying at a high angle-of-attack ([20], p.180). The roll motion of the aircraft is modelled by the equation

$$\ddot{\phi} = \vartheta_1 \phi + \vartheta_2 \dot{\phi} + \vartheta_3 |\phi| \dot{\phi} + b \delta_A \quad (12)$$

where ϕ is the roll angle, δ_A is the aileron deflection angle, $\vartheta_1 = -26.67$, $\vartheta_2 = 0.76485$, $\vartheta_3 = -2.9225$, and $b = 1.5$. As we disregard the time constant of the first-order actuator dynamics, δ_A constitutes the control input (i.e., $u = \delta_A$). After a first-order Euler's discretization of (12) with respect of the time, we obtain

$$x_{1,t+1} = x_{1t} + \Delta t x_{2t}$$

$$x_{2,t+1} = x_{2t} + \Delta t [\vartheta_1 x_{1t} + \vartheta_2 x_{2t} + \vartheta_3 x_{2t} f_m(x_{1t}) + b u_t]$$

where $x_{1t} \triangleq \phi(t\Delta t)$, $x_{2t} \triangleq \dot{\phi}(t\Delta t)$, $\Delta t = 0.01$, and $f_m(\phi) = \frac{1}{\beta} \left[\ln(2 + e^{\beta\phi} + e^{-\beta\phi}) - \ln 4 \right]$ is a differentiable function that approximates the modulus $|\phi|$ for large enough values of the parameter β (we took $\beta = 50$). Moreover, the system state is affected by perturbations producing variations of the state x_1 randomly distributed between -0.5 and 0.5 . Such perturbations are modelled as external events occurring asynchronously and affecting the system behaviour.

Without feedback control, the origin of the state plane is an unstable equilibrium point; the trajectories starting from initial points located in a neighborhood of the origin exhibit cycling oscillations typical for wing rock. The receding-horizon cost defined in this case is the following:

$$J_t = \sum_{i=t}^{t+N-1} [\rho u_i^2 + 400 (v_t^* - v_{i+1})^2], \quad t = 0, 1, \dots \quad (13)$$

where $v_i = x_{1i}$, and ρ is a positive scalar. The addressed control problem is the stabilization of the closed-loop system, with $v_t^* = 0, t \geq 0$. The control functions included in class Γ are determined as defined in Section 3 and, in particular, are characterized by $N = 5$ and by the fact that the control action is weighted by specified control weights $\rho_j, j \in I$.

7.2 Definition of the state automaton realizing the DEL

The system stabilization is here sought by exploiting the stabilizing properties of the proposed hybrid control scheme. The control objective here addressed can be also achieved by applying only one of the control functions included in Γ . The necessity of adopting a hybrid control scheme arises when observing that the control scheme can be further improved by adding a discrete-event supervisor capable of choosing the best control function to be applied among the defined set. Actually, the feedback control scheme realized by using only the CSL exhibits improvable behaviours particularly in correspondence of:

- i) large values of the state variable x_2 ; in this case, the regulator should perform in a *slower* way, thus, requiring a higher weighting factor for the control action;
- ii) small values of the state variable x_2 ; analogously, now, the control action could be *faster*, that is, the weighting factor for the control action should be lower.

The objective of the discrete-event supervisor has been identified, in this particular case, with the choice of the best weighting factor for the control action, depending on the system state and on the possible occurrence of perturbations.

The action implemented by the discrete-event supervisor is, in general, that of choosing a control function with higher ρ in case (i) and a control function with lower ρ in case (ii).

A large experimental campaign has led to a better definition of the DEL action. More specifically, it has been found that:

- the increase (decrease) of the weighting factor of the control action between two control functions subsequently applied, namely $\gamma_i, \gamma_j, i, j \in I, i \neq j$, is identified with a predefined value $\bar{\rho}$, that is,

$$\rho_j = \rho_i * \bar{\rho} \quad (\rho_j = \rho_i / \bar{\rho})$$

(a suitable choice is $\bar{\rho} = 10.0$).

- the weighting factor of the control action should be increased also when a change of sign occurs between two subsequent values of the state variable x_2 ;
- if the weighting factor of the control action has been increased (decreased) in the previous time interval, it should not be varied;
- if a perturbation in the system state has occurred, the weighting factor should be reset to its default value ($= 100.0$).

Following these reasoning lines, the state automaton realizing the DEL has been defined in the following way:

- $\tilde{E} \triangleq \{e_0, e_1\}$, where e_0 = null event, e_1 = perturbation on the system state;
- $\tilde{X} \triangleq \{\xi = \text{col}[\xi^1, \xi^2] : \xi^1 \in \{-1, 0, 1, 2\}, \xi^2 \in \{a, b, c\}\}$. The first component of the automaton state, namely ξ^1 , identifies the change occurred on the weighting factor of the control action in the preceding time interval, whereas the second component of the automaton state, ξ^2 , indicates whether the present system state satisfies specific conditions or not;
- of course, in this case, $\Sigma(\xi) = \tilde{E}, \forall \xi \in \tilde{X}$.

More specifically, the first component of the automaton state vector ξ_t at time instant t takes on the following values:

- $\xi_t^1 = -1$, which means that ρ_j has been decreased at time instant t ;
- $\xi_t^1 = 0$, which means that ρ_j has not been changed at time instant t ;
- $\xi_t^1 = +1$, which means that ρ_j has been increased at time instant t ;
- $\xi_t^1 = +2$, which means that ρ_j has been assigned to its default value ($\rho_j = 100.0$) at time instant t .

As regards the second component of the automaton state vector, it can take the following values and meanings:

- $\xi_t^2 = a$, which implies that the value of the state variable x_2 is greater than a predefined upper bound and a change of sign has occurred between the two last subsequent values of x_2 ;
- $\xi_t^2 = b$, which implies that the value of the state variable x_2 is smaller than a predefined bound and no change of sign has occurred between the two last subsequent values of x_2 ;
- $\xi_t^2 = c$, which implies that the value of the state variable x_2 is between its lower and upper bounds and no change of sign has occurred between the two last subsequent values of x_2 .

7.3 The automaton transition function

The automaton transition function defining the dynamic behaviour of the state automaton can be expressed as follows:

$$IF \quad e_{[t,t+1)} \supseteq e_1 \quad THEN$$

$$\xi_{t+1} = [2, \xi_t^2]; \quad \forall \xi_t^2$$

ELSE

$$IF \quad \xi_t^1 = +1 \quad OR \quad \xi_t^1 = -1 \quad THEN$$

$$\xi_{t+1} = [0, \xi_t^2] \quad \forall \xi_t^2$$

ELSE

$$IF \quad \xi_t^2 = a \quad THEN$$

$$\xi_{t+1} = [1, a]$$

$$IF \quad \xi_t^2 = b \quad THEN$$

$$\xi_{t+1} = [-1, b]$$

$$IF \quad \xi_t^2 = c \quad THEN$$

$$\xi_{t+1} = [0, c]$$

The automaton transition function acts by changing the value of ξ^1 in dependence of the previous values of ξ^1 and ξ^2 , and of the possible occurrence of an event of type e_1 in the preceding time interval, and according to the rules above defined.

The automaton output function updates the value of the weighting factor of the control action and chooses, as the next regulator to use, the RH regulator characterized by such a factor.

7.4 Simulation results

Some numerical results are here proposed relevant to the comparison of the behaviour of the system with application of the CSL and of the hybrid control scheme. The control problem here addressed is the stabilization of the system described in Subsection 7.1. In the considered example the initial value of the system state is $x_{10} = 0.5$ and $x_{20} = 0.0$. Moreover, asynchronous perturbations affect the system by modifying the value of the state variable x_1 with variations randomly distributed between -0.5 and 0.5 . The perturbations have been modelled by means of the event e_1 whose occurrence time is again a random variable uniformly distributed between 30 and 70 time intervals.

The set Γ of optimal feedback control functions, with reference to the receding-horizon cost given in (13), is determined as in Section 3. Such functions are characterized by $N = 5$ and, above all, are parameterized by the value of the control weight ρ . Being the aim of this section to compare the performance of the hybrid control scheme with that of an optimal feedback regulator acting only at the continuous level, it has been necessary to identify the best control function to be applied to the plant in the CSL. Such a function has been determined as $\gamma_j, j \in I, \rho_j = 100.0$, and 100.0 is also taken as the default value of the control weight to be used by the discrete-event supervisor.

The results of the above mentioned comparison between the overall hybrid control scheme and that of the CSL only are compared in Figs. 2–4. The dotted lines are referred to the behaviour of the state variables x_1 and x_2 by applying only the RH regulator characterized by $\rho_j = 100.0$ whereas the solid lines correspond to the application of the hybrid control scheme.

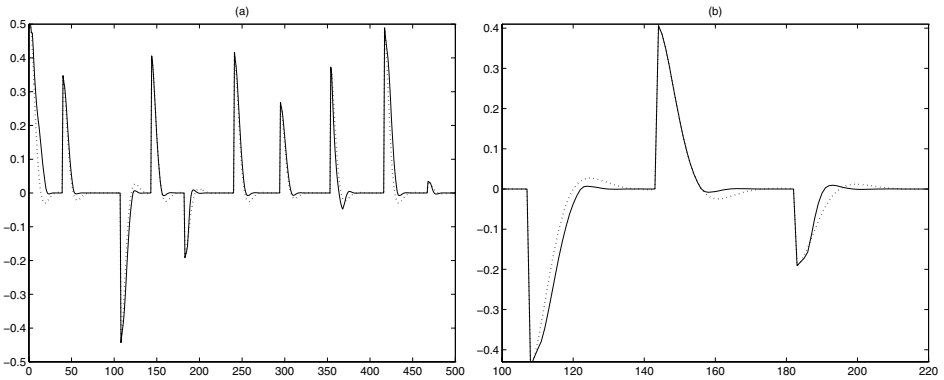


Fig. 2. (a) Behaviour of x_1 under the action of the hybrid controller (solid line) and under the action of the RH controller (dotted line). (b) Zoom of the same trajectories in the time interval $[100, 220]$.

As can be noticed from Figs. 2 and 3, the transient behaviors of the state components are significantly different in the two cases. More specifically, in most

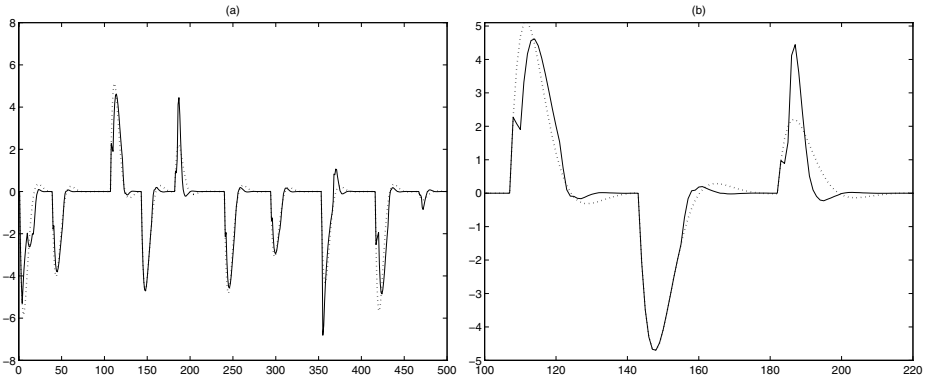


Fig. 3. Behaviour of x_2 under the action of the hybrid controller (solid line) and under the action of the RH controller (dotted line). (b) Zoom of the same trajectories in the time interval $[100, 220]$.

cases the behavior of x_1 is faster and with a smaller overshoot. Moreover, the interesting feature is that no trial-and-error procedure should be devised a-priori in order to find the best control weight, in the sense that the discrete-event supervisor is able to choose it on-line with respect to a given pre-specified criterion. In order to appreciate the effects of the control weight changes, in Fig. 4 the behavior of the control weight is plotted together with a zoom of the two behaviors of x_1 . Notice that the better transient is obtained at the price of some lack of smoothness of the state behavior. However, this effect is not surprising as hybrid systems like the one considered in the paper are in general described by discontinuous state equations [11].

8 Conclusions

The design of an hybrid control scheme for non-linear discrete-time systems is the objective of this paper. The proposed control scheme is made of a continuous level and a discrete-event supervisor. At the continuous level a set of optimal feedback control functions is determined, whereas the discrete-event level realizes the choice of the best control function, in the defined set, to be applied in each time interval.

The effectiveness of the proposed approach is first investigated by proving a preliminary result about the stability of the overall scheme, and, then, shown by means of a complete example. The stability analysis set up here goes towards the possibility of embedding the considered hybrid system into a switched system in order to exploit the same reasoning lines as in [11]. To this end, the definition of invariant sets for the considered systems and some stability concepts regarding the same invariants are already established and reported.

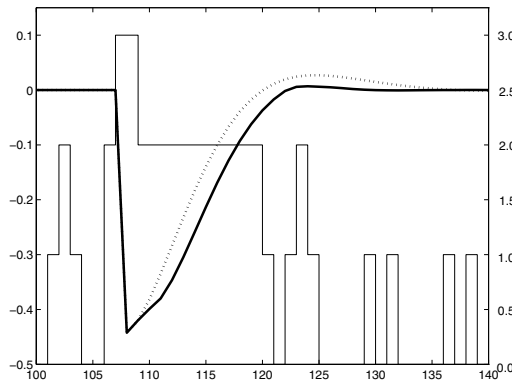


Fig. 4. Zoom of the behaviour of x_1 under the action of the hybrid controller (solid bold line) and under the action of the RH controller (dotted line). The behaviour of the control weight is also depicted (solid line). The values of x_1 and of the control weight, are reported on the left and on the right axes, respectively.

References

1. R.W. Brockett. Hybrid models for motion control systems. In H.L. Trentelman and J.C. Willems, editors, *Essays in Control*. Birkhauser, Boston, 1993.
2. A. Nerode and W. Kohn. Models for hybrid systems: automata, topologies, controllability, observability. In R.L. Grossman, A. Nerode, A.P. Ravn, and H. Rischel, editors, *Hybrid Systems*. Springer-Verlag, Berlin, 1993.
3. P.J. Antsaklis, J.A. Stiver, and M.D. Lemmon. Hybrid system modeling and autonomous control systems. In R.L. Grossman, A. Nerode, A.P. Ravn, and H. Rischel, editors, *Hybrid Systems*. Springer-Verlag, Berlin, 1993.
4. M.S. Branicky, V.S. Borkar, and S. K. Mitter. A unified framework for hybrid control. In *Proc. 33rd IEEE Conf. on Decision and Control*, pages 4228–4234, Lake Buena Vista, FL, 1994.
5. B. Lennartson, M. Tittus, B. Egardt, and S. Pettersson. Hybrid Systems in Process control. *IEEE Control Systems*, 16:45–56, 1996.
6. T. Niinomi, B.H. Krogh, and J.E.R. Cury. Synthesis of supervisory controllers for hybrid systems based on approximating automata. In *Proc. 34th IEEE Conf. on Decision and Control*, pages 1461–1466, New Orleans, LA, 1995.
7. M. Dogruel, U. Ozguner, and S. Drakunov. Sliding-mode control in discrete-state and hybrid systems. *IEEE Transactions on Automatic Control*, 41:414–417, 1996.
8. P. Varaiya. Smart cars on smart roads: Problems of control. *IEEE Transactions on Automatic Control*, 38:195–206, 1993.
9. M. D. Lemmon, C. Bett, P. Szymanski, and P. J. Antsaklis. Constructing hybrid control systems from robust linear control agents. In A. Nerode, P. Antsaklis, W. Kohn, and S. Sastry, editors, *Hybrid Systems II*. Springer-Verlag, Berlin, 1995.
10. M. Branicky. Stability of switched and hybrid systems. In *Proc. 33rd IEEE Conf. on Decision and Control*, pages 3498–3503, Lake Buena Vista, FL, 1994.

11. Ling Hou, A.N. Michel, and Hui Ye. Stability analysis of switched systems. In *Proc. 34th IEEE Conf. on Decision and Control*, pages 1208–1212, New Orleans, LA, 1995.
12. Hui Ye, A.N. Michel, and Ling Hou. Stability theory for hybrid dynamical systems. In *Proc. 34th IEEE Conf. on Decision and Control*, pages 2679–2684, New Orleans, LA, 1995.
13. J. A Stiver, P. J. Antsaklis, and M. D. Lemmon. Hybrid control system design based on natural invariants. In *Proc. 34th IEEE Conf. on Decision and Control*, pages 1455–1460, New Orleans, LA, 1995.
14. J.A. Stiver, P.J.Antsaklis, and M.D. Lemmon. An invariant based approach to the design of hybrid control systems. In *Proc. 13th IFAC Triennial World Congress*, pages 467–472, San Francisco, CA, 1996.
15. M. Lemmon and C. Bett. Robust hybrid control system design. In *Proc. 13th IFAC Triennial World Congress*, pages 395–400, San Francisco, CA, 1996.
16. S.S. Keerthi and E.G. Gilbert. Optimal infinite-horizon feedback laws for a general class of constrained discrete-time systems: stability and moving-horizon approximations. *Journal of Optimization Theory and Applications*, 57:265–293, 1988.
17. T. Parisini and R. Zoppoli. A receding-horizon regulator for nonlinear systems and a neural approximation. *Automatica*, 31:1443–1451, 1995.
18. D.Q. Mayne and H. Michalska. Receding Horizon Control of Nonlinear Systems. *IEEE Trans. on Automatic Control*, 35:814–824, 1990.
19. T. Parisini, M. Sanguineti, and R. Zoppoli. Nonlinear stabilization by receding-horizon neural regulator. In *Proc. 34th IEEE Conf. on Decision and Control* (also to appear in the *Int. Journal of Control*), pages 2433–2441, New Orleans, LA, 1995.
20. M. Krstić, I. Kanellakopoulos, and P. Kokotović. *Nonlinear and Adaptive Control Design*. Wiley, New York, 1995.

Approximating Automata and Discrete Control for Continuous Systems – Two Examples from Process Control

Jörg Raisch^{1*}, Eberhard Klein², Siu O’Young^{3 **}, Christian Meder², and Alexander Itigin²

¹ Max-Planck-Institut für Dynamik komplexer technischer Systeme
Leipziger Str. 44
D-39120 Magdeburg, FR Germany
raisch@isr.uni-stuttgart.de, raisch@mpi-magdeburg.mpg.de

² Institut für Systemdynamik und Regelungstechnik
Universität Stuttgart
Pfaffenwaldring 9
D-70550 Stuttgart, FR Germany
{klein, meder, itigin}@isr.uni-stuttgart.de

³ Faculty of Engineering and Applied Science
Memorial University of Newfoundland
St. John’s, Newfoundland, Canada A1B 3X5
oyoung@engr.mun.ca

Abstract. A standard problem in hybrid control systems theory is to design discrete, or symbolic, feedback for a given continuous plant. When specifications are discrete, this problem can be solved by first approximating the continuous plant model by a (nondeterministic) automaton, and then synthesizing discrete (supervisory) control for the automaton. A necessary condition is that the approximation behaviour contains the behaviour of the underlying continuous plant model. Then, any controller forcing the approximation to obey the specifications will also force the continuous model to satisfy the specifications. We use a version of this approach which allows adjustment of approximation accuracy to address two simple process control problems: supervisory control of a three-tank laboratory experiment and safety enforcement for an evaporator. In both cases, the entire design process is carried through: we first determine a suitable abstraction, compute the minimally restrictive supervisor, and then present examples for closed loop trajectories.

1 Introduction

This contribution deals with an approximation based approach to hybrid control systems design and its application to two specific examples – supervisory control

* Support from Deutsche Forschungsgemeinschaft under Grant Ra 516/3-1 and through “Sonderforschungsbereich” SFB 412 is gratefully acknowledged.

** Research supported by the Information Technology Research Centre and Condata Technologies Ltd., both of Ontario, Canada.

of a three-tank laboratory experiment and safety enforcement for an evaporator. This paper builds on previous work by the first and third author [11,12,13]. Its purpose is twofold: on the theoretical side, the accuracy of the approximation step is increased by “trimming” the transition structure suggested in [12,13]; this improvement follows directly from the input/output view adopted in the paper. On the practical side, it is demonstrated that approximation based approaches are indeed feasible for a class of small but realistic hybrid control problems.

Using discrete approximations, or abstractions, to design discrete feedback for continuous plants is an appealing idea: such an approximation based approach “translates” potentially difficult hybrid problems into purely discrete problems, which can subsequently be solved using tools from DES (Discrete Event Systems) theory. The approach constitutes a rigorous design method, if the approximation behaviour (the set of all pairs of input/output signals that are compatible with the approximation equations) contains the behaviour of the “underlying” continuous plant model. Then, any controller that solves the problem on the discrete level (forces the approximation to obey a given set of specifications) will also force the continuous plant model to satisfy the specifications.

Approximating continuous systems by discrete models for the purpose of feedback design has been suggested by *Antsaklis*, *Lemmon*, and *Stiver* (e.g. [17,1]), and by *Lunze* (e.g. [9]). In the original version of their approach, approximation accuracy is completely determined by measurement quantization; the measurement map $q_y : \mathbb{R}^n \rightarrow Y_d$ (where Y_d is a finite set) partitions the plant state space $\mathbb{R}^n$, and the “cells” induced by this partition correspond to the abstraction states. However, approximation accuracy is a crucial parameter for control systems design: if the discrete approximation is “too coarse”, no suitable controller exists, and the problem cannot be solved unless one can change the granularity of measurement quantization (this phenomenon will be illustrated for the three-tank example). Hence, one needs an abstraction scheme which, for a given measurement map, allows approximation accuracy to be adjusted to various specifications. Such a scheme has been suggested in [11,12,13]: it provides a totally ordered set of discrete abstractions, where ordering is in the sense of approximation accuracy. The approximation step in this scheme can be further improved by adopting an input/output point of view as described in [14].

This paper is organized as follows: in Section 2, it is explained how to build discrete abstractions for a continuous model by first approximating its behaviour and then generating realizations for the approximate behaviours. Section 3 summarizes a procedure for synthesizing minimally restrictive supervisors for the discrete abstractions. This synthesis method is a modification of *Ramadge’s* and *Wonham’s* supervisory control theory (e.g. [15,16]); it can be directly applied to the discrete models obtained in the approximation step. It has been described in [13]; we include a short summary to make the present paper self-contained. In Sections 4 and 5, we use our approach to design supervisory control for a three-tank laboratory experiment and a safety enforcement scheme for an evaporator.

Finally, a remark concerning notation and terminology: signals are interpreted as maps; the domain of a signal is time (denoted by T), its codomain is

referred to as the “signal space”. Time can be continuous ($T = \mathbb{R}^+$) or discrete ($T = \{t_0, t_1, \dots\}$). As most of our paper is set in a discrete-time framework, we use the same symbol for signals in continuous and in discrete time. The important distinction is between continuous-valued and discrete-valued signals: the codomains of the former are dense subsets of Euclidean space, the latter “live” (take values) in discrete sets. Signals are represented by lower case letters, their codomains by the corresponding upper case letters. Discrete-valued signals are characterized by the subscript “d”. For example, $y_d : \{t_0, t_1, \dots\} \rightarrow Y_d$ is a discrete-valued measurement signal which is defined on the sampling grid $\{t_0, t_1, \dots\}$ and “lives” in the (discrete) set Y_d . The codomains of all discrete-valued signals are assumed to be finite, their elements (the possible values the signal can take at each instant of time) are characterized by superscripts: the i -th element in the set Y_d , for example, is denoted by $y_d^{(i)}$. Finally, irrespective of the nature of time, we call a plant model “continuous” (“discrete”), if its state variable is continuous-valued (discrete-valued).

2 A Hierarchy of Discrete Abstractions for a Given Continuous System

In this section, we introduce an ordered set of discrete abstractions for a given continuous plant model. We will first describe the continuous model, then specify the behaviours that the discrete abstractions are required to exhibit; then, in a subsequent step, we will come up with minimal realizations for these behaviours.

2.1 The Continuous Plant Model

The continuous “base” model of the plant is described by the (vector) differential equation

$$\frac{dx}{dt}(t) = \tilde{f}(x(t), u_d(t)), \quad (1)$$

where $t \in \mathbb{R}^+$ denotes time and $x(t) \in X \subseteq \mathbb{R}^n$ is the continuous state at time t . $u_d(t) \in U_d$ is the control input, with U_d being a finite set of symbols:

$$U_d = \{u_d^{(1)}, \dots, u_d^{(\alpha)}\}.$$

We work in a discrete-time framework with fixed sampling grid $T = \{t_0, t_1, t_2, \dots\} \subset \mathbb{R}^+$. Hence, the control signal is piecewise constant and can only change value for $t = t_k$, $k = 0, 1, \dots$. Likewise, measurement information is only available at the sampling instants. Hence, we switch to a discrete-time version of the continuous plant model,

$$x(t_{k+1}) = f(x(t_k), u_d(t_k)), \quad (2)$$

$$y_d(t_k) = q_y(x(t_k)), \quad (3)$$

where $f(x(t_k), u_d(t_k))$ is the solution of (1) at time t_{k+1} for initial condition $x(t_k)$ and control input $u_d(t) = u_d(t_k)$, $t_k \leq t < t_{k+1}$ (the solution $f(x, u_d)$ is assumed to exist for all $x \in X$ and $u_d \in U_d$). The measurement signal $y_d : T \rightarrow Y_d$ “lives” on a finite set of symbols

$$Y_d = \{y_d^{(1)}, \dots, y_d^{(\gamma)}\}.$$

$f : X \times U_d \rightarrow X$ is the state transition map, $q_y : X \rightarrow Y_d$ the measurement map. Without loss of generality, the latter is required to be onto.

2.2 Behaviours

Let $T = \{t_0, t_1, \dots\}$ be the chosen sampling grid. Denote the set of all functions from T into $(U_d \times Y_d)$ by $(U_d \times Y_d)^T$. Then, the (discrete-time) behaviour of the continuous plant model, $\mathcal{B}_c \subseteq (U_d \times Y_d)^T$, is the set of all pairs of control and measurement signals which are compatible with the model equations (2), (3). For a survey on “behavioural” systems theory see [19].

Clearly, a necessary condition for *any* discrete abstraction is that its behaviour $\mathcal{B}_l$ must contain the discrete-time behaviour $\mathcal{B}_c$ of the underlying continuous model:

$$\mathcal{B}_c \subseteq \mathcal{B}_l \tag{4}$$

implies that every sequence of input/measurement symbols that the continuous plant model can generate, can also be produced by the discrete approximation. If this condition were violated, the continuous system could respond to a given input signal with an unacceptable measurement signal which would not be predictable by the discrete approximation. Hence, this unacceptable phenomenon could not be suppressed by a control strategy based on the discrete approximation – the approximation would be useless as far as control systems design is concerned. As illustrated in Figure 1, this “abstraction condition” is also sufficient for the purposes of control systems design:

suppose we design a causal (discrete) feedback controller (with input signal y_d and output signal u_d) and hook it up to the discrete abstraction. Obviously, the controller behaviour $\mathcal{B}_{controller}$ is also a subset of $(U_d \times Y_d)^T$; the feedback loop consisting of controller and abstraction exhibits behaviour $\mathcal{B}_l \cap \mathcal{B}_{controller}$ – only those pairs of input/output signals “survive” that are compatible with both abstraction and controller equations. From Figure 1, it is immediately clear that the “abstraction condition” (4) implies:

$$\mathcal{B}_l \cap \mathcal{B}_{controller} \subseteq \mathcal{B}_{spec} \implies \mathcal{B}_c \cap \mathcal{B}_{controller} \subseteq \mathcal{B}_{spec}; \tag{5}$$

in other words: if the controller forces the discrete approximation to obey the specifications, the feedback loop consisting of discrete controller and continuous “base” model will also meet the specifications.

It is worth while pointing out that “blocking” ($\mathcal{B}_c \cap \mathcal{B}_{controller} = \emptyset$) cannot occur within the proposed framework: as we work on a fixed sampling grid T ,

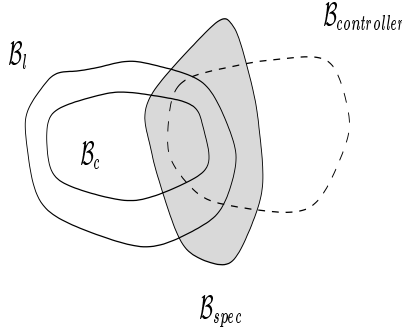


Fig. 1. Visualization of “abstraction condition”.

“blocking” would imply that connecting controller and continuous plant model would, literally, stop time. In mathematical terms, “blocking” cannot occur within our framework, as any (discrete-time) feedback loop composed of a strictly causal system (the continuous plant model (2), (3)) and a causal system (the controller) has a solution trajectory on T .

If (4) holds, the “size” of the difference $B_l \setminus B_c$ is an indicator for the accuracy of the discrete approximation: the “smaller” $B_l \setminus B_c$, the smaller the loss in “prediction power” when replacing the continuous model by its discrete abstraction. The trivial abstraction (characterized by the fact that, regardless of previous control inputs, every measurement symbol can occur at each sampling instant) has no “prediction power” whatsoever. It is obvious, that no controller can enforce the specifications on the abstraction level, if the plant approximation is “too coarse” (if, for example, the trivial abstraction is chosen).

2.3 Abstraction Behaviours

In order to specify desired abstraction behaviours, we need a bit of additional notation (sorry): recall that $T = \{t_0, t_1, \dots\}$ is the sampling grid. Denote the intervals $\{t_0, \dots, t_k\}$ and $\{t_{k+1}, \dots\}$ by T_k and T_{k+} , respectively. If t_k refers to the present sampling instant, T_{k+} comprises “the future”, and T_k “the past and present”. Define $B_c^k \subseteq (U_d \times Y_d)^{T_k}$ to be the restriction of the behaviour B_c to the interval T_k :

$$B_c^k := \{b^k \in (U_d \times Y_d)^{T_k} \mid \exists b^{k+} \in (U_d \times Y_d)^{T_{k+}} \text{ such that } [b^k, b^{k+}] \in B_c\}.$$

Finally, introduce the *predicted output set* $\mathcal{Y}_c(b^k)$ as the set of all measurement symbols that the continuous model (2), (3) can possibly generate at time t_{k+1} if the string b^k has occurred. Then, the continuous system behaviour can be written iteratively as :

$$B_c^{k+1} = \left\{ \left[b^k, \left(y_d^{(j)}, u_d^{(i)} \right) \right] \mid b^k \in B_c^k, u_d^{(i)} \in U_d, y_d^{(j)} \in \mathcal{Y}_c(b^k) \right\}. \quad (6)$$

Now, we define less accurate prediction sets $\mathcal{Y}_l$, $l = 0, 1, \dots$, by using only “recent data”. By “recent data” (denoted by $b^{k,l}$), we mean the restriction of b^k to the interval

$$T_{k,l} := \begin{cases} \{t_{k-l}, \dots, t_k\} & \text{if } k > l, \\ \{t_0, \dots, t_k\} & \text{if } k \leq l. \end{cases}$$

Hence, $b^{k,l}$ represents a string of control and measurement symbols reaching back to time t_{k-l} or, if $k \leq l$, to the initial sampling instant t_0 . Obviously,

$$b^k = \begin{cases} [b^{k-l-1}, b^{k,l}] & \text{if } k > l, \\ b^{k,l} & \text{if } k \leq l. \end{cases}$$

$\mathcal{Y}_l(b^{k,l})$ is defined as the set of all measurement symbols that the continuous model (2), (3) can generate at time t_{k+1} if the string $b^{k,l}$ has occurred during the time interval $T_{k,l}$. Clearly,

$$\mathcal{Y}_l(b^{k,l}) := \begin{cases} \bigcup_{b^{k \otimes l \otimes 1}} \mathcal{Y}_c(b^k) & \text{if } k > l, \\ \mathcal{Y}_c(b^k) & \text{if } k \leq l, \end{cases}$$

i.e. $\mathcal{Y}_l(b^{k,l})$ is obtained as the union of the prediction sets of all data strings b^k which coincide on the interval $T_{k,l}$ (but may differ during the “distant past” T_{k-l-1}) (see Figure 2 for an illustration). Therefore,

$$\mathcal{Y}_0(b^{k,0}) \supseteq \mathcal{Y}_1(b^{k,1}) \supseteq \dots \mathcal{Y}_l(b^{k,l}) \dots \supseteq \mathcal{Y}_c(b^k). \quad (7)$$

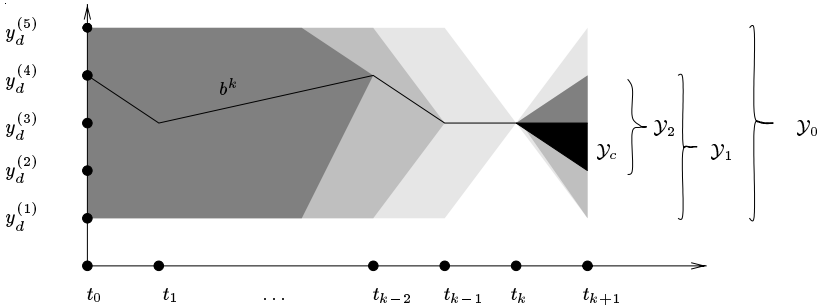


Fig. 2. Predicted output sets.

From the prediction sets $\mathcal{Y}_l$, the approximation behaviours $\mathcal{B}_l$ can be defined iteratively by

$$\mathcal{B}_l^0 := \mathcal{B}_c^0, \quad (8)$$

$$\mathcal{B}_l^{k+1} := \left\{ \left[b^k, \left(y_d^{(j)}, u_d^{(i)} \right) \right] \mid b^k \in \mathcal{B}_l^k, u_d^{(i)} \in U_d, y_d^{(j)} \in \mathcal{Y}_l(b^{k,l}) \right\}, \quad k = 0, 1, \dots \quad (9)$$

From (6), (7), (8), and (9), it follows immediately that

$$\mathcal{B}_0 \supseteq \mathcal{B}_1 \supseteq \dots \mathcal{B}_l \dots \supseteq \mathcal{B}_c. \quad (10)$$

Now that we have specified the abstraction behaviours $\mathcal{B}_l$, $l = 0, 1, \dots$, we only need to explain how to find discrete realizations (state models) that generate these behaviours, or, equivalently, the corresponding output prediction sets $\mathcal{Y}_l(b^{k,l})$.

2.4 Realization of Abstraction Behaviour $\mathcal{B}_l$

In this section we show how to build a discrete state model A_l with predicted output set $\mathcal{Y}_l(b^{k,l})$ and, consequently, behaviour $\mathcal{B}_l$. We first select the *state set* of A_l .

Recall that, by definition, the state of any dynamical system summarizes all the information that, together with the current input, is needed to predict the future: Clearly, to compute the output prediction set $\mathcal{Y}_l(b^{k,l})$, we need to know $b^{k,l}$. Hence, we define the state x_d of the abstraction A_l at time t_k to consist of all the control and measurement symbols in the string $b^{k,l}$, with the exception of the current control input $u_d(t_k)$. For $l = 0$, the result is trivial:

$$x_d(t_k) := y_d(t_k);$$

for $l = 1, 2, \dots$, we get:

$$x_d(t_k) := \begin{cases} [y_d(t_0)], & \text{if } k = 0, \\ ([y_d(t_0), \dots, y_d(t_k)], [u_d(t_0), \dots, u_d(t_{k-1})]), & \text{if } k = 1, \dots, l, \\ ([y_d(t_{k-l}), \dots, y_d(t_k)], [u_d(t_{k-l}), \dots, u_d(t_{k-1})]), & \text{if } k > l. \end{cases} \quad (11)$$

This choice is reminiscent of observer canonical realizations in (continuous) control theory (e.g. [8]) – the state is made up of known quantities, and hence is trivially observable. As both U_d , the set of control symbols, and Y_d , the set of measurement symbols, are finite, the state set X_d of the discrete abstraction is also finite: an upper bound for the number of elements of X_d is given by the number of different strings (11) one gets by exhaustive permutation of measurement and control symbols.

It is obvious, however, that the continuous system (2) – (3) cannot generate *all* these strings. In the next step, one therefore introduces a “feasibility check” and eliminates all strings which are not compatible with the continuous model (2), (3): let $1 \leq \rho \leq l$ and denote $f(x, u_d^{(i)})$ by $f_i(x)$. Then, $\left([y_d^{(j_k \otimes \rho)}, \dots, y_d^{(j_k)}], [u_d^{(i_k \otimes \rho)}, \dots, u_d^{(i_k \otimes 1)}] \right)$ is an element in the state set X_d of

the discrete abstraction if and only if there exists a solution $\tilde{x} \in X$ for

$$y_d^{(j_k)} = q_y (f_{i_k \otimes 1} (f_{i_k \otimes 2} (\dots f_{i_k \otimes \rho} (\tilde{x}) \dots))) \quad (12)$$

$$y_d^{(j_k \otimes 1)} = q_y (f_{i_k \otimes 2} (\dots f_{i_k \otimes \rho} (\tilde{x}) \dots)) \quad (13)$$

$$\vdots$$

$$y_d^{(j_k \otimes \rho+1)} = q_y (f_{i_k \otimes \rho} (\tilde{x})) \quad (14)$$

$$y_d^{(j_k \otimes \rho)} = q_y (\tilde{x}) \quad (15)$$

This is a formal way of saying that there exists a value $\tilde{x}$ of the continuous state variable, such that applying the string $[u_d^{(i_k \otimes \rho)}, \dots, u_d^{(i_k \otimes 1)}]$ of control symbols makes the continuous model respond with the string $[y_d^{(j_k \otimes \rho)}, \dots, y_d^{(j_k)}]$ of measurement symbols.

This answers the question whether a string $\left([y_d^{(j_k \otimes \rho)}, \dots, y_d^{(j_k)}], [u_d^{(i_k \otimes \rho)}, \dots, u_d^{(i_k \otimes 1)}] \right)$ is an element in the state set of the discrete abstraction for $\rho = 1, \dots, l$. What about the case $\rho = 0$, i.e. strings of the form $[y_d^{(j)}]$? As we do not assume any a-priori knowledge on the continuous system state, it could be anywhere in X at time t_0 (when we “start looking at the system”). Hence, any measurement symbol in Y_d can occur at time t_0 and $X_{d0} := Y_d$ is the *set of possible initial states* of the abstraction.

Remark 1. It is clear that, in general, feasibility does not have to be checked for every combination (11) of measurement and control symbols: if (12) - (15) does not have a solution $\tilde{x}$ for a given string $\left([y_d^{(j_k \otimes \rho)}, \dots, y_d^{(j_k)}], [u_d^{(i_k \otimes \rho)}, \dots, u_d^{(i_k \otimes 1)}] \right)$, $0 < \rho < l$, any “extension”

$$\left([y_d^{(j_k \otimes \lambda)}, \dots, y_d^{(j_k \otimes \rho)}, \dots, y_d^{(j_k)}], [u_d^{(i_k \otimes \lambda)}, \dots, u_d^{(i_k \otimes \rho)}, \dots, u_d^{(i_k \otimes 1)}] \right), \quad \rho < \lambda \leq l,$$

is also non-feasible. Checking the set (12) - (15) for the existence of solutions is of course non-trivial, and, in general, one has to resort to an iterative scheme. If the transition map f of the underlying continuous model is not known (the model is given in continuous-time), each iteration step also needs to call an ODE-solver. This might seem to be an awful lot of computation. One has to keep in mind, however, that a fairly coarse approximation A_l may be sufficient to solve a given control problem. Indeed, for the examples in Section 4 and 5, abstraction A_1 proves to be adequate. Moreover, for an important special case, the feasibility check reduces to a numerically straightforward procedure, namely checking whether a set of *linear inequalities* possesses a non-empty solution set. This special case is characterized by the fact that the right hand side of (2) is affine in the state $x(t_k)$, and the measurement map has the form $q_y = Q_y \circ C_y$ (where $C_y : \mathbb{R}^n \rightarrow \mathbb{R}^p$ is a linear map, and the “quantizer” $Q_y : \mathbb{R}^p \rightarrow Y_d$ partitions $\mathbb{R}^p$ into finitely many rectangular boxes with edges parallel to the coordinate axes). The computational procedure for this case is described in [12].

This finishes the construction of the abstraction state set X_d . The number of elements in X_d is denoted by N_v :

$$X_d := \{x_d^{(1)}, \dots, x_d^{(N_v)}\}. \quad (16)$$

Up to now, we have only defined the state set of the abstraction A_l . We now describe the transition structure.

Denote the strings of control and measurement symbols associated with a particular $x_d^{(i)}$ by $u^*(x_d^{(i)})$ and $y^*(x_d^{(i)})$, respectively, and introduce a “forgetting operator” $\mathcal{F}$ which deletes the “oldest” symbol from strings $y_d^*(x_d(t_k))$ and $u_d^*(x_d(t_k))$, if $k \geq l$:

$$\begin{aligned} \mathcal{F}(y_d^*(x_d(t_k))) &:= \begin{cases} [y_d(t_0), \dots, y_d(t_k)], & \text{if } k = 0, 1, \dots, l-1 \\ [y_d(t_{k-l+1}), \dots, y_d(t_k)], & \text{if } k \geq l, \end{cases} \\ \mathcal{F}(u_d^*(x_d(t_k))) &:= \begin{cases} [u_d(t_0), \dots, u_d(t_{k-1})], & \text{if } k = 0, 1, \dots, l-1 \\ [u_d(t_{k-l+1}), \dots, u_d(t_{k-1})], & \text{if } k \geq l. \end{cases} \end{aligned}$$

Now, writing down the transition structure of the discrete abstraction is easy – at least conceptually: $(x_d^{(i)}, u_d^{(j)}, x_d^{(k)})$ is a transition in A_l if and only if

1. there exists a $y_d^{(m)} \in Y_d$ such that

$$\begin{aligned} y_d^*(x_d^{(k)}) &= [\mathcal{F}(y_d^*(x_d^{(i)})), y_d^{(m)}], \\ u_d^*(x_d^{(k)}) &= [\mathcal{F}(u_d^*(x_d^{(i)})), u_d^{(j)}], \end{aligned}$$

2. $\left([y_d^*(x_d^{(i)}), y_d^{(m)}], [u_d^*(x_d^{(i)}), u_d^{(j)}]\right)$ is compatible with the continuous system model (2) – (3).

The first condition can be verified by simple visual inspection of the abstraction states $x_d^{(i)}$ and $x_d^{(k)}$: do they “overlap” in the indicated manner? The second condition can be checked by setting up a set of equations as in (12) – (15) and testing it for the existence of solutions. Suppose both conditions hold, i.e. $(x_d^{(i)}, u_d^{(j)}, x_d^{(k)})$ turns out to be a transition in A_l . Then, $x_d^{(i)}$ and $x_d^{(k)}$ are called the exit state and the entrance state of the transition; the control symbol $u_d^{(j)}$ is its transition label. Each state $x_d^{(i)}$ has an associated unique (measured) output, which is simply the rightmost symbol in $y^*(x_d^{(i)})$. Hence, for each nonnegative integer l , we get a finite Moore automaton as a discrete abstraction for the continuous plant model. It is clear that for a given state $x_d^{(i)}$ and a given control symbol $u_d^{(j)}$, more than one $y_d^{(m)} \in Y_d$ may satisfy conditions 1 and 2: then, applying $u_d^{(j)}$ at state $x_d^{(i)}$ may drive the abstraction into more than one successor state (each carrying a different output symbol) – the resulting automaton is nondeterministic. By construction, the resulting state model is a realization of the output prediction set $\mathcal{Y}_l$.

Remark 2. In [11,12], we proposed slightly different abstractions $\tilde{A}_l$: both $\tilde{A}_l$ and A_l share the same state set, but the transition structure of $\tilde{A}_l$ is more conservative: $(x_d^{(i)}, u_d^{(j)}, x_d^{(k)})$ is declared to be a transition of $\tilde{A}_l$, if condition 1 holds. The omission of condition 2 gives less accurate approximation behaviours, but evidently does not affect their ordering. It can be argued that trimming the transition structure by adding condition 2 is a direct consequence of the input/output view adopted in [14] and in the present paper: deriving the state model A_l as a realization of the output prediction set $\mathcal{Y}_l$ naturally leads to conditions 1 and 2.

Remark 3. It is understood that the discrete abstractions evolve on the same sampling grid as the underlying continuous system: the time needed for a transition is the sampling interval $t_{i+1} - t_i$. This explains why the notion of time is trivially retained at the abstraction level.

Remark 4. As the abstraction state $x_d(t_k)$ is made up of known quantities (see equation (11)), A_l is guaranteed to be observable. It is also straightforward to prove that every element in X_d can be reached from an initial state $x_d \in X_{d0}$ (see [14] for details). Hence, A_l is a *minimal* realization of the abstraction behaviour $\mathcal{B}_l$.

3 Supervisory Control for a Discrete Abstraction

For the purposes of control systems design, we convert the Moore automaton A_l into an equivalent¹ transition structure without state outputs. This is done by adding to each state a selfloop that is labelled with the appropriate measurement symbol $y_d^{(j)}$. To ensure “correct” ordering of transitions, we compose this transition structure with the simple clock process shown in Figure 3. This also makes time explicit – the **tick** event represents the passage of one sampling interval and can be thought of as being synchronized with an external clock. The arrows labelled by U_d and Y_d represent the occurrence of any symbol from U_d or Y_d , respectively. The resulting nondeterministic transition structure M is referred to as a *Discrete-time Discrete Event Process (DTP)*. It includes explicit timing (**tick**-events) and a control mechanism (transitions labelled by a control symbol can be disabled). Its behaviour is defined as follows: count the **tick**-events; denote transitions occurring before the first **tick** by $u_d(t_0), y_d(t_0)$, transitions occurring between the first and the second **tick** by $u_d(t_1), y_d(t_1)$ etc.; in this way, the DTP generates a behaviour identical to the one of the underlying Moore automaton.

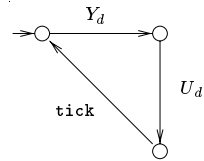


Fig. 3. Clock process.

¹ in the sense of having the same behaviour as the Moore automaton.

We adopt a supervisory control philosophy [15] to suitably modify the behaviour of the discrete plant model. We use a state-transition-based framework which is well suited to deal with nondeterminism in the discrete plant model, and which can also handle time².

The departure point is a discrete approximation of the plant model in the form of a DTP. The mechanism of control is via a subset of transitions that can be disabled (prevented from happening). These are the transitions labelled by a control symbol. Specifications are also formulated as DTPs, which may or may not (depending on whether time is important in the specifications) contain the **tick**-symbol. Forming the parallel composition $P = M \parallel S$ of the plant DTP, M , and the specification DTP, S , formally removes all transitions which violate the specifications – but this is done without caring for realizability: for example, a transition can only be eliminated if it is labelled by a control symbol; a transition cannot be eliminated if this implies that the process can reach a state where no further **tick**-event can be executed – stopping time is impossible. The optimal supervisor’s job can then be thought of as implementing the “least restrictive”, but realizable substructure of P . This is formalized in the following paragraph:

Let Q and Σ be the (finite) sets of states and event (or transition) labels of P . A transition is represented by a triplet (q_e, σ, q_i) , with $q_e, q_i \in Q$ and $\sigma \in \Sigma$. q_e and q_i are called *exit* and *entrance state*; σ is the *event label*. Transitions $(q_{e1}, \sigma_1, q_{i1})$ and $(q_{e2}, \sigma_2, q_{i2})$ are called *partners*, if $q_{e1} = q_{e2}$ and $\sigma_1 = \sigma_2$ (they have the same exit state and event label). A DTP can be represented by a pair (Δ, Q_0) where Δ and Q_0 are the (finite) sets of transitions and initial states, respectively.

Definition 1. Let $P = (\Delta, Q_0)$ be a DTP. The DTP $\tilde{P}$ modeled by the pair $(\tilde{\Delta}, \tilde{Q}_0)$ is called a Discrete-Time Discrete Event SubProcess (DSP) of P (denoted by $\tilde{P} \subseteq P$), if $\tilde{\Delta} \subseteq \Delta$, $\tilde{Q}_0 \subseteq Q_0$, and a transition $\delta \in \Delta$ can only be an element in $\tilde{\Delta}$, if all its partners are also contained in $\tilde{\Delta}$.

A state $q_2 \in Q$ is *reachable* from a state $q_1 \in Q$ (or, equivalently, q_1 is *coreachable* from q_2), if there is a sequence of transitions from Δ connecting q_1 with q_2 . A DTP P is called *reachable* if every element of its state set is reachable from an initial state.

Definition 2. Let $P = M \parallel S$ and U_d be the set of transition labels of M (and hence P) which can be disabled by a control agent. Let $\tilde{P} = (\tilde{\Delta}, \tilde{Q}_0)$ be a reachable DSP of P with state set $\tilde{Q}$ and with $\tilde{Q}_0 = Q_0$. Then $\tilde{P}$ is said to be *controllable* w.r.t. to M , if $(q_{Me}, \sigma, q_{Mi}) \in \Delta_M$ (the transition set of M), $(q_{Me}, q_{Se}) \in \tilde{Q}$, and $((q_{Me}, q_{Se}), \sigma, -) \notin \tilde{\Delta}$ implies that $\sigma \in U_d$ (– means “don’t care”).

Clearly, a DSP $\tilde{P}$ of P can only be realized by a controller if it is controllable w.r.t. M . Another condition for realizability is that the progress of time can never be stopped:

² An extension of [15] to (deterministic) DES with timing features is described in [3].

Definition 3. A state $q_1 \in Q$ is said to be **tick-coreachable** if there exists a state $q_2 \in Q$ where a **tick-transition** can occur and which is reachable from q_1 . The DTP P is called **tick-coreachable** or temporally nonblocking, if every state in Q is **tick-coreachable**.

Let $\tilde{P}_1 = (\tilde{\Delta}^1, \tilde{Q}_0^1)$ and $\tilde{P}_2 = (\tilde{\Delta}^2, \tilde{Q}_0^2)$ be two DSPs of P . Then the union of $\tilde{P}_1$ and $\tilde{P}_2$, denoted by $\tilde{P}_1 \cup \tilde{P}_2$ is a transition structure represented by the pair $(\tilde{\Delta}^1 \cup \tilde{\Delta}^2, \tilde{Q}_0^1 \cup \tilde{Q}_0^2)$. It is immediately clear that $\tilde{P}_1 \cup \tilde{P}_2$ is another DSP of P . The relation $\subseteq$ induces a partial ordering on the set of all DSPs of P . Let $\{\tilde{P}_{CN}\}$ be the set of all DSPs of P which are controllable w.r.t M and temporally nonblocking. $\{\tilde{P}_{CN}\}$ is closed under union. Hence, if non-empty, $\{\tilde{P}_{CN}\}$ forms an upper-semilattice (with the join operation being $\cup$). Clearly, $\{\tilde{P}_{CN}\}$ is finite. Therefore, the following holds:

Property 1. If $\{\tilde{P}_{CN}\}$ is non-empty, there exists a (unique) greatest DSP of P (w.r.t. the ordering via $\subseteq$) which is controllable w.r.t. M and temporally nonblocking.

If $\{\tilde{P}_{CN}\}$ is non-empty, denote its supremal element by $\tilde{P}_S$. It can be interpreted as the transition structure of P that survives under the least restrictive realizable supervisory control policy which guarantees the specifications to be met. $\tilde{P}_S$ (and hence the least restrictive control strategy) can be formally synthesized via a fixed-point algorithm in a computer-aided design environment. This procedure has been implemented in C++ with an object oriented architecture.

If $\{\tilde{P}_{CN}\}$ is empty, the supervisory control problem has no solution. This implies that either the approximating automaton A_l is too coarse (we need to provide a finer approximation A_{l+i} , $i \geq 1$), or the specifications are too strict (they cannot be met no matter how accurate our approximation is) and need to be relaxed.

Remark 5. “Forbidden state” type specifications are particularly straightforward to deal with: they are static, hence we don’t need a specification automaton. Formally, the problem can be solved as follows: certain states in the plant automaton A_l (and hence the DTP M) are declared to be forbidden. If a DSP of M does not contain any forbidden states, it is called *safe*. Then we simply check whether the set of all safe, controllable, and temporally nonblocking DSPs of M is non-empty. If so, the problem is solvable, and the supremal element in the set, denoted by $\tilde{M}_S$, is the minimally restrictive supervisory control policy.

4 A Three-tank Laboratory Experiment

4.1 Plant and Continuous Model

Our first example is a three-tank laboratory experiment (Figure 4) which can be purchased from *amira GmbH*, Duisburg. It consists of three plexiglass tanks with identical cross sectional area $A = 154\text{cm}^2$. The tanks are connected by two pipes; tank 3 is also equipped with an outlet pipe. All pipes have cross

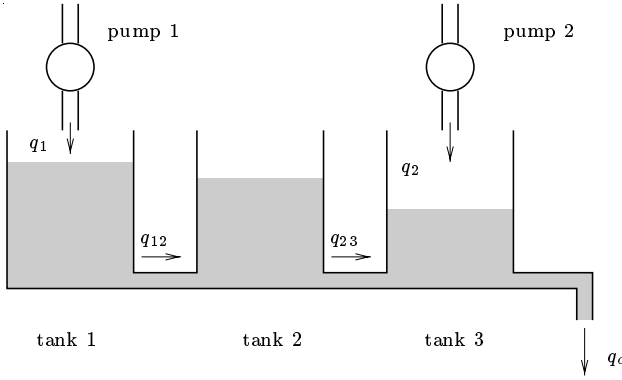


Fig. 4. Three-tank laboratory experiment.

sectional area $a = 0.5\text{cm}^2$. The pumps attached to tank 1 and 3 can be operated independently. Their flow rates are denoted by q_1 and q_2 , respectively. In both cases, the maximum flow rate is $q_{\max} = 6 \frac{\text{litres}}{\text{min}}$. Denote the water level in tank i by x_i , $i = 1, \dots, 3$. Now, we can derive a continuous model by writing down (volume) balance equations for each tank:

$$A \frac{dx_1}{dt} = q_1 - q_{12} \quad (17)$$

$$A \frac{dx_2}{dt} = q_{12} - q_{23} \quad (18)$$

$$A \frac{dx_3}{dt} = q_2 + q_{23} - q_o . \quad (19)$$

Applying Toricelli's law yields:

$$q_{12} = \begin{cases} a_{z_{12}} a \sqrt{2g} \sqrt{x_1 - x_2} & \text{if } x_1 \geq x_2 \\ -a_{z_{12}} a \sqrt{2g} \sqrt{x_2 - x_1} & \text{if } x_1 < x_2 \end{cases} \quad (20)$$

$$q_{23} = \begin{cases} a_{z_{23}} a \sqrt{2g} \sqrt{x_2 - x_3} & \text{if } x_2 \geq x_3 \\ -a_{z_{23}} a \sqrt{2g} \sqrt{x_3 - x_2} & \text{if } x_2 < x_3 \end{cases} \quad (21)$$

$$q_o = a_{z_o} a \sqrt{2g} \sqrt{x_3} , \quad (22)$$

where $g = 9.81 \frac{\text{m}}{\text{s}^2}$ is the gravitational constant. a_{z_o} , $a_{z_{23}}$, and $a_{z_{12}}$ are loss coefficients of the respective pipes. They depend on the geometry of the pipes and the water levels in the tanks. In our experimental set-up, the influence of water levels on these coefficients was found to be fairly weak; they were therefore approximated by constants:

$$a_{z_o} = a_{z_{23}} = a_{z_{12}} = 0.75 . \quad (23)$$

We choose a sampling interval of $\Delta t := t_{k+1} - t_k = 100\text{s}$ and operate the system by switching both pumps between maximum flow rate ($q_i = q_{\max}$), half

the maximum flow rate ($q_i = 0.5q_{\max}$), and “off”. Hence, we get 9 different values for the control input u_d (Table 1). We restrict ourselves to highly quantized

Table 1. Control symbols for the three-tank experiment.

$u_d^{(1)}$	$:=$	“ $q_1 = 0$ ”	&	“ $q_2 = 0$ ”
$u_d^{(2)}$	$:=$	“ $q_1 = 0$ ”	&	“ $q_2 = \frac{q_{\max}}{2}$ ”
$u_d^{(3)}$	$:=$	“ $q_1 = 0$ ”	&	“ $q_2 = q_{\max}$ ”
$u_d^{(4)}$	$:=$	“ $q_1 = \frac{q_{\max}}{2}$ ”	&	“ $q_2 = 0$ ”
$u_d^{(5)}$	$:=$	“ $q_1 = \frac{q_{\max}}{2}$ ”	&	“ $q_2 = \frac{q_{\max}}{2}$ ”
$u_d^{(6)}$	$:=$	“ $q_1 = \frac{q_{\max}}{2}$ ”	&	“ $q_2 = q_{\max}$ ”
$u_d^{(7)}$	$:=$	“ $q_1 = q_{\max}$ ”	&	“ $q_2 = 0$ ”
$u_d^{(8)}$	$:=$	“ $q_1 = q_{\max}$ ”	&	“ $q_2 = \frac{q_{\max}}{2}$ ”
$u_d^{(9)}$	$:=$	“ $q_1 = q_{\max}$ ”	&	“ $q_2 = q_{\max}$ ”

measurement information: at each sampling instant, we only “see” whether the water level in tank 3 is below 30 cm, above 60 cm, or in between:

$$y_d = \begin{cases} y_d^{(1)} & \text{if } 0 < x_3 \leq 30 \text{ cm} \\ y_d^{(2)} & \text{if } 30 \text{ cm} < x_3 \leq 60 \text{ cm} \\ y_d^{(3)} & \text{if } 60 \text{ cm} < x_3 \leq x_{\max} . \end{cases} \quad (24)$$

4.2 Specifications

We investigate two sets of specifications: the first one basically requires that the water level in tank 3 should, after a short transitional period, always be kept in the middle range. An automaton that expresses this specification is shown in Figure 5. At the first three sampling instants, any measurement symbol is allowed. After that, only $y_d^{(2)}$ may occur. There are no restrictions on the control symbols, and explicit timing (the inclusion of **tick**-symbols) is not required. Hence, the set of transition labels is Y_d .

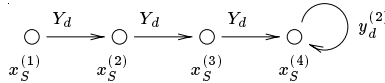


Fig. 5. Specification automaton S_d .

A slightly more demanding specification is the following. After a start-up period of three sampling intervals (where we allow any measurement symbol),

$y_d^{(2)}$ has to occur at the next ten sampling instants. Then, after a transitional period of two sampling intervals, we want the water level in tank 3 to drop below 30 cm (the symbol $y_d^{(1)}$ has to occur once), before the cycle starts again. This specification is captured in the automaton shown in Figure 6. Again, the set of transition labels in the specification automaton is Y_d .

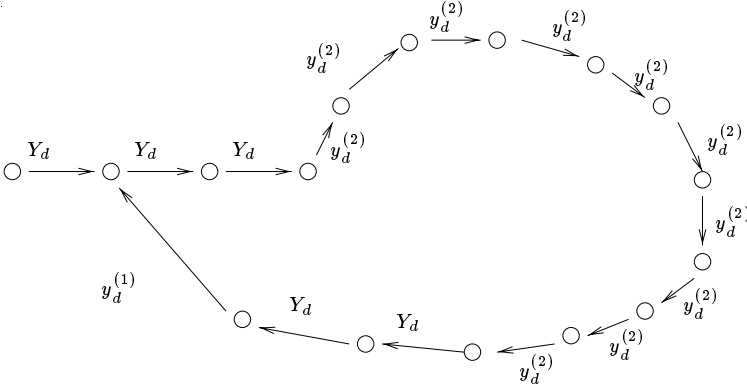


Fig. 6. Specification automaton S_b .

4.3 Discrete Abstractions and Control

With the algorithm suggested in Section 2, we can now compute the abstractions A_l , $l = 0, 1, \dots$. Recall that A_0 is the coarsest element in our set of approximations. The Moore automaton A_0 consists of only three states – the measurement symbols $y_d^{(1)}$, $y_d^{(2)}$, and $y_d^{(3)}$ – and it turns out to have 64 transitions. We now apply the synthesis method outlined in Section 3 to check whether the two control problems defined above are solvable. Denote the DTP corresponding to A_0 by M_0 and form the synchronous products $P_{0a} := M_0 \parallel S_a$ and $P_{0b} := M_0 \parallel S_b$ (the specification automata S_a and S_b are shown in Figure 5 and Figure 6, respectively). Not surprisingly, the set of controllable and temporally nonblocking DSPs turns out to be empty for both P_{0a} and P_{0b} : on this (coarse) level of abstraction, neither of the two problems can be solved.

Hence, we need a more accurate discrete approximation and compute the Moore automaton A_1 . It consists of 67 states and 962 transitions. Denote the corresponding DTP by M_1 and form $P_{1a} := M_1 \parallel S_a$. Now, the set of all controllable and temporally nonblocking DSPs of P_{1a} is non-empty. Its supremal element $\tilde{P}_S$ realizes the least restrictive supervisory control scheme; it has 36 states and 123 transitions. The supervisor action can be inferred from a table that lists possible control inputs for each of these states – an excerpt is shown in Table 2: if, for example, only one measurement symbol has been observed

Table 2. Possible control symbols (excerpt).

supervisor state state of A_1 state of S_a				possible control symbols
$y_d(t_{k-1})$	$y_d(t_k)$	$u_d(t_{k-1})$		
-	$y_d^{(1)}$	-	$x_S^{(2)}$	$u_d^{(9)}$
			$\vdots$	
$y_d^{(1)}$	$y_d^{(2)}$	$u_d^{(9)}$	$x_S^{(3)}$	$u_d^{(3)}$
$y_d^{(1)}$	$y_d^{(2)}$	$u_d^{(9)}$	$x_S^{(3)}$	$u_d^{(9)}$
			$\vdots$	
$y_d^{(2)}$	$y_d^{(2)}$	$u_d^{(3)}$	$x_S^{(4)}$	$u_d^{(3)}$
			$\vdots$	

($t_k = t_0$), if it indicates that the water level in tank 3 is “low” ($y_d(t_k) = y_d^{(1)}$), and if the specification automaton S_a is in state $x_S^{(2)}$, the only possible control action is $u_d^{(9)}$ (“switch both pumps to full flow rate”). If, on the other hand, the observed plant state is $([y_d^{(1)}, y_d^{(2)}], [u_d^{(9)}])$ and S_a is in state $x_S^{(3)}$, the supervisor allows the control symbols $u_d^{(3)}$ and $u_d^{(9)}$; any one of them can be picked by an arbitrary selection mechanism (if there are several options, we always choose the first line)– both of them are guaranteed not to violate the specification.

Figure 7 illustrates how the control scheme works: it shows a closed loop

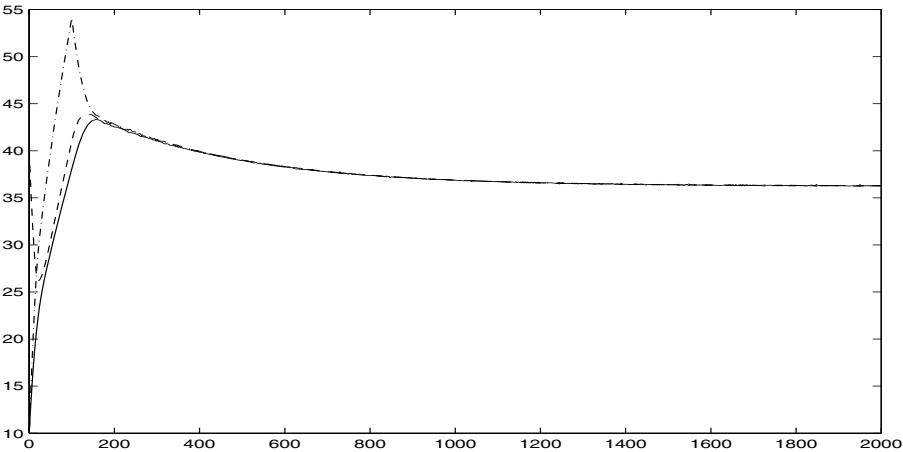


Fig. 7. Simulation of closed loop ($x_1(t)$ is dashdotted, $x_2(t)$ dashed, $x_3(t)$ solid).

simulation of the continuous plant model (17) – (24) under discrete supervisory control. We start with an initial condition $x(0) = [10\text{cm } 40\text{cm } 10\text{cm}]$. Hence, the first available measurement information is $y_d(t_0 = 0) = y_d^{(1)}$ (i.e. water level in tank 3 is low). The first occurrence of a measurement symbol drives S_a into state $x_S^{(2)}$. The supervisor responds with control symbol $u_d^{(9)}$. The water level starts to rise in tank 1 and 3, and the next available measurement information (at time $t_1 = 100\text{s}$) is $y_d^{(2)}$. The new plant state is $([y_d^{(1)}, y_d^{(2)}], [u_d^{(9)}])$, and S_a has switched to state $x_S^{(3)}$. Accordingly, the control input is changed to $u_d^{(3)}$, and the plant again responds with $y_d(t_2) = y_d^{(2)}$, etc.

We apply the same procedure for the second specification: the set of controllable and temporally nonblocking DSPs of $P_{1b} := M_1 \parallel S_b$ is also non-empty. Its supremal element has 86 states and 251 transitions. Figure 8 shows a simulation of the continuous plant model (17) – (24) under discrete control; the initial conditions for the simulation are $x(0) = [10\text{cm } 80\text{cm } 50\text{cm}]'$. Again, as in the previous case, the controller consists of the supervisor $\tilde{P}_S$ and an arbitrary mechanism that picks any one of the control symbols that $\tilde{P}_S$ allows. Clearly, the closed loop satisfies the specifications.

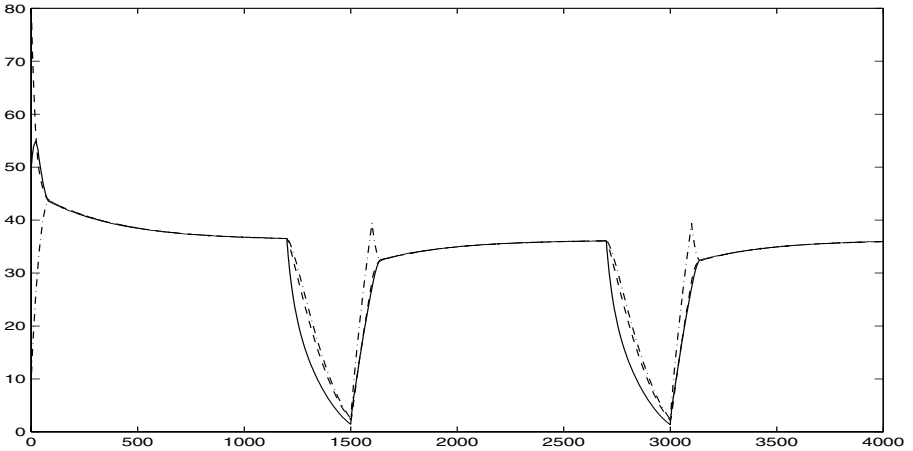


Fig. 8. Simulation of closed loop ($x_1(t)$ is dashdotted, $x_2(t)$ dashed, $x_3(t)$ solid).

5 Safety Enforcement for an Evaporator

5.1 Plant and Continuous Model

The second example is an evaporator for a sodium chloride solution (Figure 9).

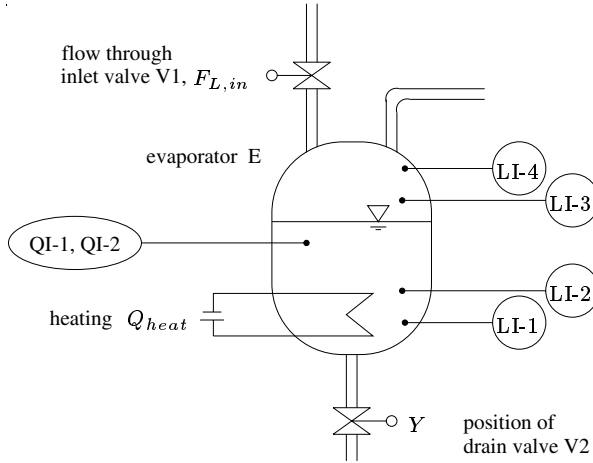


Fig. 9. Evaporator.

It has been suggested in [7] as a benchmark problem for the synthesis of discrete event supervisors and sequential logic controllers in the process industries. The process is meant to be run in batch mode: the evaporator is filled, salt concentration is increased to a desired level by boiling, then the evaporator is emptied, and the cycle starts again. We only consider the safety enforcement part for this problem. Safety enforcement prohibits any situation which is regarded to be unsafe or undesirable and hence provides a (minimally restrictive) framework, within which a lower level (sequential logic) controller operates and forces the system into the appropriate batch cycle.

As state variables, we choose molar holdup (n_L [mol]), molar concentration of sodium chloride in the liquid phase (x_s [mol/kmol]), and temperature (T [K]). Hence, the state set X of the continuous model is a dense subset of $\mathbb{R}^3$. Our continuous model for the evaporator is based on balance equations:

1. total material balance [mol/s]:

$$\frac{dn_L}{dt} = F_{L,in} - F_L - F_V, \quad (25)$$

2. component material balance of sodium chloride [mol/(kmol s)]:

$$\frac{dx_s}{dt} = \frac{1}{n_L} (F_{L,in}(x_{s,in} - x_s) + F_V x_s), \quad (26)$$

3. energy balance [J/s]:

$$\begin{aligned}
 c_{p,L} n_L M_L \frac{dT}{dt} &= F_{L,in} (h_{L,in} - h_L) - F_V \Delta h \\
 &+ \frac{\partial \Delta h}{\partial x_s} (F_{L,in} (x_{s,in} - x_s) + F_V x_s) \\
 &+ Q_{heat} - \alpha (A^v + \frac{\pi D^v n_L M_L}{A^v \rho_L}) (T - T_{amb}) . \quad (27)
 \end{aligned}$$

Liquid and vapour flow (F_L and F_V) are auxiliary variables, $x_{s,in}$, $h_{L,in}$, α , A^v , D^v , and T_{amb} constant parameters, M_L , $c_{p,L}$, h_L , Δh and ρ_L physical properties. All of them are explained in the Appendix.

Heating power Q_{heat} [W], position Y [–] of the drain valve, and molar liquid flow $F_{L,in}$ [mol/s] are control inputs. They are all operated in an “on/off” manner:

$$\begin{aligned}
 Q_{heat} &\in \{0, 6000W\} \\
 Y &\in \{0, 1\} \\
 F_{L,in} &\in \{0, 2mol/s\}.
 \end{aligned}$$

The latter variable corresponds to the position of the inlet valve V_1 , its possible values can be realized by opening or closing V_1 . If we exclude combinations where both valves are open (this doesn’t make physical sense), we have six different control symbols, denoted by $u_d^{(1)} \dots u_d^{(6)}$ (Table 3):

Table 3. Control symbols for the evaporator problem.

$u_d^{(1)}$	:= “valve V_1 closed” & “valve V_2 closed” & “heating off”
$u_d^{(2)}$	:= “valve V_1 closed” & “valve V_2 closed” & “heating on”
$u_d^{(3)}$	:= “valve V_1 closed” & “valve V_2 open” & “heating off”
$u_d^{(4)}$	:= “valve V_1 closed” & “valve V_2 open” & “heating on”
$u_d^{(5)}$	:= “valve V_1 open” & “valve V_2 closed” & “heating off”
$u_d^{(6)}$	:= “valve V_1 open” & “valve V_2 closed” & “heating on”

We sample the plant at intervals of 20s, i.e. the control signal is changed and measurement information is available only at the sampling instants $t_0 = 0, t_1 = 20s, t_2 = 40s, \dots$. Measurement information consists of quantized versions of liquid volume $V_L = \frac{n_L}{\rho_L} M_L$ and salt concentration x_s . This is illustrated in Figure 10.

5.2 Specifications

In this example, specifications can be easily expressed in terms of measurement and control symbols:

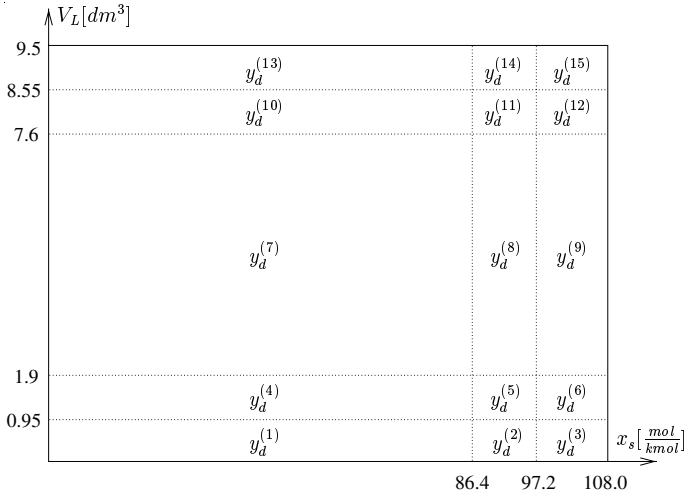


Fig. 10. Measurement symbols.

Spec. 1: the heating must not be switched “on” if $V_L \leq 0.95$; this translates into $u_d(t_k) \notin \{u_d^{(2)}, u_d^{(4)}, u_d^{(6)}\}$ if $y_d(t_k) \in \{y_d^{(1)}, y_d^{(2)}, y_d^{(3)}\}$.

Spec. 2: overflow has to be avoided, i.e. $V_L \leq 8.55$ or, equivalently, $y_d(t_k) \notin \{y_d^{(13)}, y_d^{(14)}, y_d^{(15)}\}$.

Spec. 3: crystallization must not occur, i.e. $x_s \leq 97.2$ or, equivalently $y_d(t_k) \notin \{y_d^{(3)}, y_d^{(6)}, y_d^{(9)}, y_d^{(12)}, y_d^{(15)}\}$.

5.3 Discrete Abstractions and Control

As in the three-tank example, abstraction A_0 is too coarse for the control problem at hand. Hence, we compute A_1 . It has 236 states and 2196 transitions. Recall that abstraction states are simply collections of measurement and control symbols. Hence, with all the specifications being static, they can be conveniently translated into “forbidden state specifications”: Spec 1, for example, attaches the label “forbidden” to all states of the form $x_d = \left(\left[y_d^{(i)}, y_d^{(j)} \right], \left[u_d^{(k)} \right] \right)$, $i = 1, 2, 3$, $k = 2, 4, 6$. Now, A_1 is converted into the corresponding DTP M_1 ; the set of all safe (recall that “safe” means that all “forbidden” states have been removed), controllable, and temporally nonblocking DSPs of M_1 is nonempty. Its supremal element $\tilde{M}_S$ corresponds to the maximally permissive safety enforcement scheme. It has 75 states, and a small excerpt is shown in Table 4. If the plant (and hence the supervisor) is in state $\left(\left[y_d^{(7)}, y_d^{(7)} \right], \left[u_d^{(1)} \right] \right)$, all possible control actions are deemed to be safe and therefore allowed. If the observed plant state

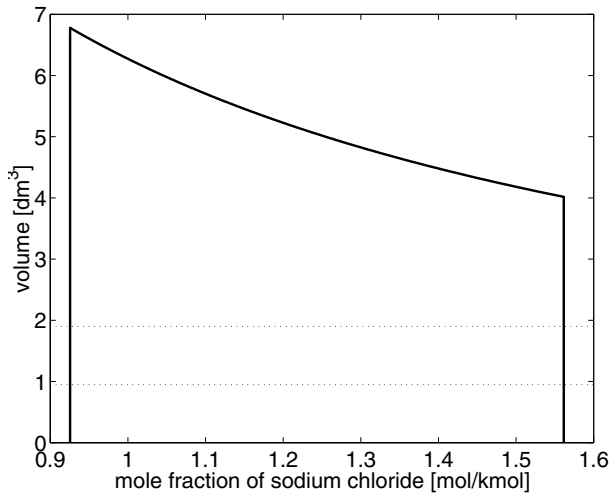


Fig. 11. Batch cycle for the evaporator system under discrete supervision

tion of the *Ramadge/Wonham* framework. Our synthesis software proved to be extremely efficient, and computation time was negligible.

References

1. P. J. Antsaklis, J. A. Stiver, and M. Lemmon. Hybrid system modelling and autonomous control systems. In [6], pages 366-392.
2. P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, editors. *Hybrid Systems Iv*, Lecture Notes in Computer Science, Vol. 1273. Springer-Verlag, 1997.
3. B. A. Brandin and W. M. Wonham. Supervisory control of timed Discrete Event Systems. *IEEE Trans. Automat. Contr.*, vol. 39, pp. 329-342, 1994.
4. J. C. Chou. Thermodynamic Properties of Aqueous Sodium Chloride Solutions from 32 to 350 F. Ph. D Thesis, Oklahoma State University, 1968.
5. K. Feldkamp. Der Wärmeübergang beim Sieden von wässrigen Lösungen. Ph. D Thesis, Technische Universität Carolo-Wilhelmina, Braunschweig, 1969.
6. R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, editors, *Hybrid Systems*, Lecture Notes in Computer Science, Vol. 736. Springer-Verlag, 1993.
7. H.-M. Hanisch and S. Kowalewski. Diskontinuierlicher Misch- und Trennprozeß, Fallbeispiel für die Modellierung ereignisdiskreter Systeme und den systematischen Entwurf von Steuerungen. In: 3. Workshop GMA-Fachausschuß 1.8: Methoden der Steuerungstechnik. 1994.
8. T. Kailath, *Linear Systems*. Prentice-Hall, 1980.
9. J. Lunze. Qualitative modelling of linear dynamical systems with quantized state measurements. *Automatica*, 30:417-431, 1994.
10. I. N. Maksimova. Determination of the Density of Aqueous Solutions. *Russian Journal of Physical Chemistry*, 39 (3): 296-297, 1965.

11. J. Raisch. Nondeterministic automata as approximations for continuous systems – an approach with an adjustable degree of accuracy. Proc. 2nd MATHMOD (International Symposium on Mathematical Modelling), pp. 195–202. IMACS, Vienna. 1997.
12. J. Raisch and S. O’Young. A Totally Ordered Set of Discrete Abstractions for a given Hybrid or Continuous System. In [2], pp. 342–360.
13. J. Raisch and S. D. O’Young. Discrete approximation and supervisory control of continuous systems. Accepted for publication in IEEE Transactions on Automatic Control, Special Issue on Hybrid Systems.
14. J. Raisch. A Hierarchy of Discrete Abstractions for a given Hybrid Plant Model. Proc. ADPM’98 (Automatisation des Processus Mixtes: les Systèmes Dynamiques Hybrides), Reims, March 1998.
15. P. J. Ramadge and W. M. Wonham. Supervisory control of a class of discrete event systems. *SIAM J. Control and Optimization*, 25:206–230 1987.
16. P. J. Ramadge and W. M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77:81–98, 1989.
17. J. A. Stiver and P. Antsaklis. Modeling and analysis of hybrid control systems. In *Proc. 31st IEEE Conference on Decision and Control*, 1992.
18. Verein Deutscher Ingenieure (VDI) – Gesellschaft Verfahrenstechnik und Chemieingenieurwesen (GVC) (ed.) *VDI-Wärmeatlas*. VDI-Verlag, Frankfurt, 5. th edition, 1988.
19. J. C. Willems. Paradigms and puzzles in the theory of dynamical systems. *IEEE Transactions on Automatic Control*, 36:259–294, 1991.
20. I. D. Zaytsev and G. G. Aseyev Properties of Aqueous Solutions of Electrolytes. CRC Press, Boca Raton, 1992.

A Details of the Evaporator Model

In this section, we specify all the details for the evaporator model. We list equations for auxiliary variables and physical property correlations (Sections A.1 and A.2), and collect all parameters in Section A.3.

A.1 Equations for Auxiliary Variables

Vapour and liquid flow (F_V [$\frac{mol}{s}$], F_L [$\frac{mol}{s}$]) are determined by

$$F_V = \frac{A^t}{v_V} \sqrt{\frac{4 \cdot 10^5 R T (p - p_{amb})}{M_w (p + p_{amb})}}, \quad (28)$$

$$F_L = (-\xi_0 + \sqrt{\xi_0^2 + 4 \xi_1 Y(p - p_{amb} + \frac{\rho_L g V_L}{A^v 10^4})}) / (2 \xi_1), \quad (29)$$

where pressure p [$\frac{N}{dm^2}$] is given by

$$p = \max(p_{amb}, p_{s,L}). \quad (30)$$

v_V (molar volume of the vapour phase) and $p_{s,L}$ (saturation pressure of sodium chloride solution) are additional physical properties (see section A.2).

A.2 Physical Property Correlations

The following physical property correlations have been used (throughout this section, x_S is in $\frac{mol}{mol}$):

- Molecular weight of sodium chloride solution $M_L [\frac{g}{mol}]$:

$$M_L = x_s (M_s - M_w) + M_w. \quad (31)$$

- The density of sodium chloride solution ($\rho_L [\frac{g}{dm^3}]$) is determined according to [10]:

$$\rho_L = \rho_w(T_R) + \sum_{i=1}^6 k_{\rho_w,i} (T - 273.15)^i + \sum_{j=1}^2 k_{\rho_s,j} x_s^j. \quad (32)$$

The constants in (32) were fitted using data from [18] and [20]:

$$\begin{aligned} \rho_w(T_R) &= 999.858, & k_{\rho_w,1} &= 0.600919 \times 10^{-1}, \\ k_{\rho_w,2} &= -0.919335 \times 10^{-2}, & k_{\rho_w,3} &= 0.133038 \times 10^{-3}, \\ k_{\rho_w,4} &= -0.21617 \times 10^{-5}, & k_{\rho_w,5} &= 0.197676 \times 10^{-7}, \\ k_{\rho_w,6} &= -0.70207 \times 10^{-10}, & k_{\rho_s,1} &= -2241.66, \\ k_{\rho_s,2} &= 2190.21. \end{aligned}$$

- Specific heat capacity of sodium chloride solution $c_{p,L} [\frac{J}{gK}]$ [4]:

$$c_{p,L} = \sum_{i=0}^2 \left(\sum_{j=0}^2 k_{c_p,ij} x_s^j \right) T^i. \quad (33)$$

$$\begin{aligned} k_{c_p,00} &= 0.136538 \times 10^1, & k_{c_p,01} &= -0.8959483 \times 10^1, \\ k_{c_p,02} &= 0.2380725 \times 10^2, & k_{c_p,10} &= -0.2032837 \times 10^{-2}, \\ k_{c_p,11} &= 0.3627181 \times 10^{-1}, & k_{c_p,12} &= -0.6216819 \times 10^{-1}, \\ k_{c_p,20} &= 0.3221832 \times 10^{-5}, & k_{c_p,21} &= -0.6152962 \times 10^{-4}, \\ k_{c_p,22} &= 0.1055741 \times 10^{-3}. \end{aligned}$$

- Molar enthalpy of vapourization of sodium chloride solution $\Delta h [\frac{J}{mol}]$ [5]:

$$\Delta h = M_L \sum_{i=0}^4 k_{\Delta h,i0} \left(\frac{M_s x_s}{M_L} \right)^i \quad (34)$$

$$\begin{aligned} k_{\Delta h,00} &= 0.2256215 \times 10^4, & k_{\Delta h,10} &= -0.1860691 \times 10^3, \\ k_{\Delta h,20} &= 0.5003167 \times 10^4, & k_{\Delta h,30} &= -0.2713489 \times 10^5, \\ k_{\Delta h,40} &= 0.5114429 \times 10^5. \end{aligned}$$

- Molar enthalpy of vapour phase of water $h_V [\frac{J}{mol}]$:

$$h_V = \sum_{i=0}^3 k_{h_V,i} (T - 273.15)^i. \quad (35)$$

$$k_{h_V,0} = 45757.0, \quad k_{h_V,1} = 32.89359, \\ k_{h_V,2} = 5.768308 \times 10^{-3}, \quad k_{h_V,3} = -7.886424 \times 10^{-8}.$$

- Molar enthalpy of liquid phase of sodium chloride solution h_L [$\frac{J}{mol}$]:

$$h_L = h_V - \Delta h. \quad (36)$$

- Saturation pressure of sodium chloride solution $p_{s,L}$ [$\frac{N}{dm^2}$] [4]:

$$p_{s,L} = 10^3 e^{(1+k_{s,s,1}x_s+k_{s,s,2}x_s^{1.5})(k_{s,w,1}-\frac{k_{s,w,2}}{T+k_{s,w,3}})+(k_{s,s,3}+k_{s,s,4}x_s^{1.5}+k_{s,s,5}x_s^2)}. \quad (37)$$

$$k_{s,w,1} = 0.23478 \times 10^2, \quad k_{s,w,2} = 0.39849 \times 10^4, \\ k_{s,w,3} = -0.39724 \times 10^2, \quad k_{s,s,1} = -0.33204656 \times 10^{-1}, \\ k_{s,s,2} = 0.13635246, \quad k_{s,s,3} = -0.19203616 \times 10^1 \\ k_{s,s,4} = 0.32750375 \times 10^1, \quad k_{s,s,5} = -0.19866438 \times 10^2.$$

- Molar volume of ideal gas v_V [dm^3/mol]:

$$v_V = \frac{RT}{p}. \quad (38)$$

A.3 Parameters

To completely specify the evaporator model, we need a number of parameters. They are listed in Table 5.

Table 5. Parameters of evaporator model.

cross sectional area of vapour tube	$A^t = 0.022698 \, dm^2$
cross sectional area of evaporator	$A^v = 3.14159 \, dm^2$
diameter of evaporator	$D^v = 2 \, dm$
gravitational constant	$g = 98.1 \, \frac{dm}{s^2}$
pressure drop coefficient	$\xi_0 = 0.8892 \, \frac{N \, s}{dm^2 \, mol}$
pressure drop coefficient	$\xi_1 = 3.23712 \, \frac{N \, s^2}{dm^2 \, mol^2}$
molecular weight of water	$M_w = 18.01488 \, \frac{g}{mol}$
molecular weight of sodium chloride	$M_s = 58.4428 \, \frac{g}{mol}$
gas constant	$R = 8.3145 \, \frac{J}{mol \, K}$
heat transfer coefficient	$\alpha = 0.02 \, \frac{W}{dm^2 \, K}$
ambient temperature	$T_{amb} = 293.15 \, K$
ambient pressure	$p_{amb} = 1013.0 \, \frac{N}{dm^2}$
molar enthalpy of inlet flow	$h_{L,in} = 7243.543 \, \frac{J}{mol}$
mole fraction of sodium chloride in inlet flow	$x_{s,in} = 0.925545 \, \frac{mol}{kmol}$

Action Systems with Continuous Behaviour

Mauno Rönkkö¹ and Anders P. Ravn²

¹ Dept. of Computer Science, Turku Centre for Computer Science
Åbo Akademi University, Lemminkäisenkatu 14A, FIN-20520 Turku, Finland
`mronkko@abo.fi`

² Department of Information Technology, Technical University of Denmark
Bldg. 344, DK-2800 Lyngby, Denmark
`apr@it.dtu.dk`

Abstract. An action system framework is a predicate transformer based method for modelling and analysing distributed and reactive systems. The actions are statements in Dijkstra's guarded command language, and their semantics is given by predicate transformers. We extend conventional action systems with a *differential action* consisting of a differential equation and an evolution guard. The semantics is given by a weakest liberal precondition transformer, because it is not always desirable that differential actions terminate. It is shown that the proposed differential action has a semantics which corresponds to a discrete approximation when the discrete step size goes to zero.

The extension gives action systems the power to model real-time clocks and continuous evolutions within hybrid systems. In this paper we give a standard form for such a hybrid action system. We also extend parallel composition to hybrid action systems. This does not change the original meaning of the parallel composition, and therefore ordinary action systems compose in parallel with hybrid action systems.

1 Introduction

Action systems, originally proposed by Back and Kurki-Suonio [3], are predicate transformer based systems for modelling discrete computations. For a given program, the predicate has the program variables as free variables. The transformer uses Dijkstra's idea of describing the effect of a given program by calculating the precondition which guarantees that a given postcondition holds. Action systems have been extensively used in sequential program refinement, and also in reactive and concurrent programs, see e.g. [4,9].

Previously action systems have been extended with explicit clock variables when reasoning about real-time systems, and with discrete approximations when modelling continuous phenomena in hybrid settings [13,19]. However, since there is no technical reason why predicates over program variables should not speak of continuous entities, it seems possible to simplify reasoning in these applications by introducing the differential equations governing evolution of clocks and other continuous variables directly into the framework. Such an extension is the main result of this paper. A guideline in developing the differential action is a wish to

preserve the connection between discrete accumulation and continuous evolution, such that the weakest liberal precondition semantics corresponds to the one found for the approximating discrete accumulation.

Inspired by the Branicky's unified model for hybrid systems [6], we identify a class of action systems suitable for modelling hybrid systems. These hybrid action systems alternate between discrete changes and continuous evolutions.

The parallel composition for ordinary action systems is described by interleaving of actions from the constituent systems. This model is based on the assumption that the effect of two simultaneous actions is the same as the effect of the two actions in some sequence. This assumption is not valid in a continuous parallel world, where the total effect of simultaneous differential actions is the sum of their individual effects. This form of composition is captured by a linear composition. We extend the parallel composition for hybrid action systems so, that it uses linear composition for combining the continuous evolutions, but interleaves discrete actions. We retain the usual meaning of the parallel composition for ordinary action systems. Hence, the mixed parallel composition between action systems and hybrid action systems remain also meaningful.

Related work. Action systems are similar to the UNITY programs [10]. Especially, the refinement methodology is central in both of these approaches. In [11] is a discussion of how a program logic may be used to reason about continuous systems. That paper focuses on exploring the potentials of refinement reasoning for such systems, whereas we concentrate on a precise definition of the semantics.

Action systems have an operational flavour, and our extension is thus related to various hybrid automata models [1,20,18] and the general hybrid model surveyed in [7]. However, our interest is not in the basic model, but in an integration of specifications (predicates) and actions such that one can reason without reference to the semantics. A similar concept is found in the description language used in HyTech [15,16], where the approximations formulated in [17] would be seen as action refinements. In applications, it seems promising to first use action refinement and standard mathematical analysis tools to weaken complex predicates with differential equations and then translate to HyTech and use its potential for automated checking.

A directly related piece of work is the duration calculus semantics for hybrid CSP [23,14], which, however, does not include a proof theory. The current paper may be seen as an investigation of a proof theory analogous to the one investigated for duration calculus in [22]. However, that work does not include parallel composition.

Overview. We start by defining action systems in Section 2. In Section 3 we derive from the weakest liberal precondition for a discrete accumulation a predicate describing continuous evolution. Based on this predicate we define the differential action. In Section 4 we study a gate example [21] to illustrate the similarities and the differences between formulations using discrete accumulation and continuous evolution. In Section 5 we identify a suitable form to describe hybrid

systems. We call such action systems hybrid action systems. In Section 6 we investigate the parallel composition of action systems and extend it so that it is meaningful also when composing hybrid action systems. In Section 7 we use a water tank example to illustrate the extended parallel composition. We also use the example to outline, how HyTech can be used in analysing hybrid action systems. Section 8 concludes and discusses directions for further work.

2 Actions

An action is any statement in Dijkstra's guarded command language [12], including pure guarded commands. In a system, the actions operate on a fixed set of variables. The values of these variables identify a state in the system. Therefore, predicates or conditions on these variables provide a convenient way of specifying and reasoning about states in a system.

When we consider a given action A , we may speak of a *postcondition* q . It is a predicate specifying desirable states after the execution of the action. By using the postcondition as a reference, we can define the meaning or the semantics for the action A . We can, for example, use the *weakest precondition* predicate transformer $\text{wp}(A, q)$ for this purpose. It returns a predicate describing the largest set of states from which the action terminates and reaches the postcondition.

The weakest precondition can also be used for calculating two basic properties for any action. These properties are the termination set and the enabledness set. The set of states from which an action terminates, i.e. reaches an arbitrary state, is given by $\text{t}A = \text{wp}(A, \text{true})$. Similarly, the set of states in which an action is enabled, i.e. the states from which an action starts to execute, is given by $\text{g}A = \neg \text{wp}(A, \text{false})$.

In the following we give the semantics for the actions with the *weakest liberal precondition*, wlp . Unlike the weakest precondition, it does not require termination. The relation between these two predicate transformers is given by Dijkstra [12] as $\text{wp}(A, q) = \text{wlp}(A, q) \wedge \text{t}A$ or equivalently when A is *conjunctive* $\text{wlp}(A, q) = (\text{t}A \Rightarrow \text{wp}(A, q))$.

Furthermore, we consider two actions A and B to be the same when their semantics agree for any postcondition in the sense of weakest liberal precondition:

$$A \equiv B \quad \text{iff} \quad \forall q \cdot \text{wlp}(A, q) = \text{wlp}(B, q)$$

Any two actions for which this relation holds can be substituted for each other in any context without affecting the meaning.

2.1 Semantics for actions

In this paper we use capital letters such as X , Y , E , and F to denote a vector or list of variables and other components. For clarity, we do not use subscripts when referring to the components in a vector; we explicitly say which small letters denote vector components, whenever this is needed. Furthermore, let

$q[E/X]$ denote textual substitution of expressions E for the free variables X in a predicate q .

Elementary actions are defined as in [5,8].

$$\begin{aligned} \text{wlp}(\mathbf{abort}, q) &\hat{=} \text{true} \\ \text{wlp}(\mathbf{skip}, q) &\hat{=} q \\ \text{wlp}(X := E, q) &\hat{=} q[E/X] \end{aligned}$$

Note that *abort* never terminates, whereas a **skip** or an assignment are sure to terminate.

Let, as before, A and B be actions, and p be a predicate. The sequential composition, the composition by a non-deterministic choice, as well as prefixing an action with a guard are defined by

$$\begin{aligned} \text{wlp}(A; B, q) &\hat{=} \text{wlp}(A, \text{wlp}(B, q)) \\ \text{wlp}(A \parallel B, q) &\hat{=} \text{wlp}(A, q) \wedge \text{wlp}(B, q) \\ \text{wlp}(p \rightarrow A, q) &\hat{=} p \Rightarrow \text{wlp}(A, q) \end{aligned}$$

An assertion ensures that a given condition p holds in the current state. If the given condition does not hold, the assertion aborts. Hence, the meaning for the assertion is $\{p\} \equiv p \rightarrow \mathbf{skip} \parallel \neg p \rightarrow \mathbf{abort}$. From which we calculate the semantics:

$$\begin{aligned} \text{wlp}(\{p\}, q) &= \text{wlp}(p \rightarrow \mathbf{skip}, q) \wedge \text{wlp}(\neg p \rightarrow \mathbf{abort}, q) \\ &= (p \Rightarrow \text{wlp}(\mathbf{skip}, q)) \wedge \text{true} \end{aligned}$$

resulting in the semantics $\text{wlp}(\{p\}, q) = (p \Rightarrow q)$.

Iteration has the liberal semantics

$$\text{wlp}(\mathbf{do} A \mathbf{od}, q) \hat{=} \forall n \geq 0 \cdot \text{wlp}(A^n, \text{g}A \vee q)$$

where $A^0 \hat{=} \mathbf{skip}$ and $A^{n+1} \hat{=} A; A^n$. The semantics for iteration is useful when proving some general properties related to the iteration of actions. For example, we can easily show that if p is an invariant for the iterated action, $p \Rightarrow \text{wlp}(A, p)$, it is also an invariant throughout the entire iteration, $p \Rightarrow \text{wlp}(\mathbf{do} A \mathbf{od}, p)$. An iteration terminates when it is no longer enabled, $t(\mathbf{do} A \mathbf{od}) = \exists n \geq 0 \cdot \neg \text{g}A^n$.

2.2 Action systems

An action system $\mathcal{A}$ is an initialised block of the form

$$\begin{aligned} \mathcal{A} = & \llbracket \mathbf{var} X : T \\ & X := E; \\ & \mathbf{do} A_1 \parallel \dots \parallel A_n \mathbf{od} \\ & \rrbracket : Z \end{aligned}$$

The expression $\mathbf{var} X : T$, declares the list of variables X with types T . Any of these variables can be *global*. This is denoted by a star (e.g. x^*) in the declaration. A global variable can be shared with other action systems. The global variables that do not belong to X , but are used by the actions $A_1 \dots A_n$, are in the imported global variables Z . The variables in X that are not global, are *local* for

the block. The local variables do not exist outside the action system where they are defined. All the local and global variables form the *state space* of an action system.

The action $X := E$ initialises the variables X by expressions E . Therefore, the global variables are initialised in the action system, where they are declared. Because the initialisation is a multiple assignment, the expression E may not refer to any variable in X or Z . For example, an initialisation $x, y := x + 3, y + 4$, where $\{x, y\} \in X$, is illegal, since the values of the variables are undefined before the initialisation. All action systems initialise before any action is taken in one of them. This becomes apparent later, when the parallel composition for the action systems is defined. After initialisation, the actions $A_1 \dots A_n$ are repeatedly executed when enabled. The selection of an enabled action for execution is non-deterministic. There are no fairness assumptions about selection of actions. The execution of an elementary action is always atomic. This ensures that during the execution, the values of the variables remain unchanged, unless the action itself changes them. If two enabled actions refer to disjoint variables, their execution can be in any order or in parallel. Hence, this models parallelism by interleaving.

The execution of the action system terminates when none of the actions is enabled, i.e. $\neg g(A_1 \parallel \dots \parallel A_n)$. Similarly, the execution of the action system aborts when an executed action aborts.

3 Differential Action

We extend the conventional set of actions with a *differential action* that describes continuous behaviour. In the following we derive the semantics for the differential action from a discrete accumulation that approximates the continuous behaviour. We also show some of the useful properties related to the use of differential actions.

3.1 Discrete accumulation

In discrete accumulation a variable x is incremented from an initial value until a guard ceases to hold (if ever). When a discrete accumulation is used to approximate a continuous behaviour, the increment $f(x)$ is approximately the differential multiplied by a fixed, positive time step h . Both the increment $f(x)$ and the guard e may depend on h . The discrete accumulation is captured by

$$\mathbf{do} \ e \rightarrow x := x + f(x) \cdot h \ \mathbf{od}$$

The meaning of the discrete accumulation is given by the weakest liberal precondition for the iteration action. Let $p[v/x]^i$ denote the i times repeated substitution $p[v/x] \dots [v/x]$, and n and m be natural numbers. By unfolding the precondition for the action above we get

$$\begin{aligned} & \text{wlp}(\mathbf{do} \ e \rightarrow x := x + f(x) \cdot h \ \mathbf{od}, \ q) \\ &= \forall n \geq 0 \cdot (e \vee q)[x + hf(x)/x]^n \vee \exists 0 \leq m < n \cdot \neg e[x + hf(x)/x]^m \end{aligned} \tag{1}$$

Analysis of this predicate is laborious, because it requires the calculation of a predicate after repeated substitutions. Fortunately, we can express the repeated substitution as a single substitution with the help of a recursive function, Φ_h , which for the fixed $h > 0$ is given by:

$$\Phi_h(n \cdot h) = x + h \cdot \sum_{i=h}^{n \cdot h} f(\Phi_h(i - h))$$

This function has the property that $\Phi_h(n \cdot h)$ equals $x[x + hf(x)/x]^n$. The function Φ_h also has an implicit form:

$$\Phi_h(0) = x, \quad \forall i \geq 0 \cdot \frac{\Phi_h(i \cdot h + h) - \Phi_h(i \cdot h)}{h} = f(\Phi_h(i \cdot h)) \quad (2)$$

It is this form that we use in the concise semantics for the discrete accumulation. From the precondition expression (1) above we obtain

$$\begin{aligned} & \text{wlp}(\mathbf{do} \ e \rightarrow x := x + f(x) \cdot h \ \mathbf{od}, \ q) \\ &= \exists \Phi_h \cdot \Phi_h(0) = x \wedge \forall i > 0 \cdot \frac{\Phi_h(i \cdot h + h) - \Phi_h(i \cdot h)}{h} = f(\Phi_h(i \cdot h)) \wedge \\ & \quad (\forall n \geq 0 \cdot (e \vee q)[\Phi_h(n \cdot h)/x] \vee \exists 0 \leq m < n \cdot \neg e[\Phi_h(m \cdot h)/x]) \end{aligned} \quad (3)$$

3.2 Evolution

In a continuous evolution, the variable x is assumed to be a differentiable function of time. If a solution exists to the differential equation $\dot{x} = f(x)$, then it is the limit of the step functions $(\Phi_h \mid h > 0)$ as h approaches zero. We denote this limit by Φ , and by applying the definition of differentiation to the formula (2), we have

$$\Phi(0) = x, \quad \dot{\Phi} = f(\Phi)$$

This observation leads us to define a new action for continuously evolving variables, called a *differential action*. The differential action evolves continuously the values of given variables X as long as a guard e holds, this guard condition may use the variables X . The evolution is controlled by a system of differential equations. We use the standard convention of writing a system of differential equations as $\dot{X} = F(X)$. The $\dot{X}$ is a component-wise first derivative of X .

The exploration above is summarised in the following definition.

Definition 1. *Let e be a guard and q be a postcondition, both may speak of the variables X . The differential action $e : \dot{X} = F(X)$ has the semantics*

$$\begin{aligned} & \text{wlp}(e : \dot{X} = F(X), \ q) \triangleq \\ & \exists \Phi \cdot \Phi(0) = X \wedge \dot{\Phi} = F(\Phi) \wedge \\ & \quad (\forall \tau \geq 0 \cdot (e \vee q)[\Phi(\tau)/X] \vee \exists 0 \leq \delta < \tau \cdot \neg e[\Phi(\delta)/X]) \end{aligned}$$

In the semantics, the first two conjuncts require the existence of the solution Φ for the differential equation $\dot{X} = F(X)$. The last term isolates the first point in time, where the guard ceases to hold.

It is difficult to capture the exact link between discrete accumulation and the differential action in the general case; and one cannot expect it to be easy, because it would answer many questions of numerical analysis. However, as an indication of the kind of result one may have, we give the following theorem.

Theorem 1. *If the discrete accumulation function Φ_h , given by*

$$\Phi_h(n \cdot h) = x + h \cdot \sum_{i=h}^{n \cdot h} f(\Phi_h(i - h))$$

converges to a solution Φ for the differential equation $\dot{x} = f(x)$ from above, i.e., $\|\Phi_h\| \leq \|\Phi\|$ for all h , and if the evolution guard e is upward closed, i.e., $\|a\| \leq \|b\|$ implies $e[a/x] \geq e[b/x]$; that is if e is true for some value, then it is true for any larger value.

Then differential termination: $\neg wlp(e : \dot{x} = f(x), false)$ implies accumulation termination: $\neg wlp(\mathbf{do} \ e \rightarrow x := x + f(x) \cdot h \ \mathbf{od}, false)$.

The implication holds in the reverse direction if ‘from below’ and ‘downward closed’ are substituted for ‘from above’ and ‘upward closed’.

3.3 Stability

If the evolution guard e remains true in a differential action, it never terminates, as seen from $wlp(e : \dot{X} = F(X), false) = true$; but the behaviour is bounded by the guard. In this case, the system becomes *stable* with respect to the guard. When something is known about the stability of the differential equation, it may be used in the analysis of the differential action.

Theorem 2. *Assume that the differential equation $\dot{X} = F(X)$ has solutions. Let O be a point in the domain of X . Let a sphere around O with radius ϵ , be denoted by $S(\epsilon)$.*

Assuming that a solution with initial value X_0 in $S(\rho)$ is Ljapunov stable in $S(\epsilon)$, then we have

$$wlp(\{X \in S(\rho)\}; X \in S(\epsilon) : \dot{X} = F(X), false) = true$$

Furthermore, if the solution is asymptotically stable, then for any $\gamma > 0$:

$$wlp(\{X \in S(\rho)\}; X \in S(\epsilon) \setminus S(\gamma) : \dot{X} = F(X), X \in S(\gamma)) = true$$

Also, if the solution for $\dot{X} = F(X)$ is periodic and O is any point on the solution curve,

$$wlp(\{X \neq O\}; X \neq O : \dot{X} = F(X), X = O) = true$$

3.4 Clocks

A clock or timer that measures the passage of time is specified with a differential action. An accurately running timer c is given by $e : \dot{c} = 1$. If it had a skew of at most 1 %, it would be $\{0.99 < \delta < 1.01\}$; $e : \dot{c} = 1 \cdot \delta$. The assertion in front of the differential action ensures that the skew is within the given limits.

A delay for four time units is given by $c := 1; \ c < 5 : \dot{c} = 1$. The corresponding discrete accumulation would have $f = 1$:

$$\mathbf{do} \ c < 5 \rightarrow c := c + 1 \cdot h \ \mathbf{od}$$

and have the accumulation function: $\Phi(n \cdot h) = c + h \cdot \sum_{i=h}^{n \cdot h} 1$ which reduces to $\Phi(n \cdot h) = c + h \cdot n$; as one would expect. It will terminate when $n = \lceil \frac{5-c}{h} \rceil$, i.e., not later than the corresponding differential action, as predicted by the second part of Theorem 1.

3.5 Composite evolutions

The following is an example of an evolution of several continuous variables. There are two variables x and v that evolve. The x variable is dependent on the v .

$$\begin{aligned} & \text{wlp}(e : \dot{x}, \dot{v} = v, 1, q) \\ &= \exists \Phi_x, \Phi_v \cdot \Phi_x(0) = x \wedge \Phi_v(0) = v \wedge \dot{\Phi}_x = \Phi_v \wedge \dot{\Phi}_v = 1 \wedge \\ & (\forall \tau \geq 0 \cdot (e \vee q)[\Phi_x(\tau), \Phi_v(\tau)/x, v] \vee \exists 0 \leq \delta < \tau \cdot \neg e[\Phi_x(\delta), \Phi_v(\delta)/x, v]) \end{aligned}$$

If we assume that the action has been preceded by an initialisation, say $x, v := 0, 0$, the clock example allows us to simplify the result to

$$\begin{aligned} & \exists \Phi_x \cdot \Phi_x(0) = 0 \wedge \dot{\Phi}_x = v \wedge \\ & (\forall \tau \geq 0 \cdot (e \vee q)[\Phi_x(\tau), \tau/x, v] \vee \exists 0 \leq \delta < \tau \cdot \neg e[\Phi_x(\delta), \delta/x, v]) \end{aligned}$$

In an application, the timer v is probably there to limit the evolution of the time varying system x . This could result in an guard $e = (v < 5)$, for which we would get:

$$\exists \Phi_x \cdot \Phi_x(0) = 0 \wedge \dot{\Phi}_x = v \wedge q[\Phi_x(5), 5/x, v]$$

Notice that $q[\Phi_x(5), 5/x, v]$ specifies all those states, from which x evolves to values $\Phi_x(5)$, when v evolves linearly to value 5.

3.6 Algebraic Properties

In the following we assume that the systems of differential equations used in the differential actions have solutions. Also, the equivalences are in the sense of weakest liberal precondition as defined in Section 2.

A differential action acts as a skip action when the evolution guard does not hold initially. This is captured by the following equivalence, which can eliminate a redundant differential action.

$$\{\neg e\}; e : \dot{X} = F(x) \equiv \{\neg e\}; \text{skip} \quad (4)$$

At termination of a differential action the evolution guard does not hold anymore. This is captured by the following equivalence, which is used for introducing a termination assertion.

$$e : \dot{X} = F(x) \equiv e : \dot{X} = F(x); \{\neg e\} \quad (5)$$

With the help of the two properties above, it is easy to prove that sequential composition of a differential action with itself has the same effect as the differential action by itself.

$$\begin{aligned} & e : \dot{X} = F(x); e : \dot{X} = F(x) \\ & \equiv \{ \text{termination assertion introduction (5)} \} \\ & e : \dot{X} = F(x); \{\neg e\}; e : \dot{X} = F(x) \\ & \equiv \{ \text{redundancy elimination (4)} \} \\ & e : \dot{X} = F(x); \{\neg e\}; \text{skip} \\ & \equiv \{ \text{skip elimination} \} \\ & e : \dot{X} = F(x); \{\neg e\} \\ & \equiv \{ \text{termination assertion removal (5)} \} \\ & e : \dot{X} = F(x) \end{aligned}$$

This illustrates that sequential composition takes no time; there is no evolution over the composition.

A differential action does not change the values of variables that do not belong to X . Therefore, we can weaken or strengthen the evolution guard by moving parts to or from an assertion. Let g be a predicate not containing variables of X , then the following holds for a differential action

$$\{g\}; g \wedge e : \dot{X} = F(x) \equiv \{g\}; e : \dot{X} = F(x)$$

4 A Gate Example

The following example illustrates the differences and similarities between a discrete and a continuous approach in modelling and analysing a system. We use the lowering of a gate as an example. The gate is lowered with a speed proportional to its current position, that is $\dot{x} = -x$. The task is to model the movement, and measure the time for the gate to become closed. The gate is started 1 meter above the stop and it is considered to be closed, when it is at most 1 cm away from the stop. Figure 1 illustrates the setting.

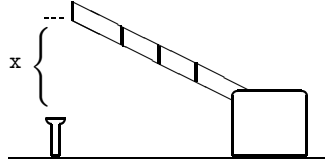


Fig. 1. The gate.

4.1 With discrete accumulation

Let, as before, h be the time step. With discrete accumulation the model is

$$x, c := 1, 0; \text{ do } x > 0.01 \rightarrow x, c := x - x \cdot h, c + 1 \cdot h \text{ od}$$

The time consumed is given by c , which is $n \cdot h$, where n is the number of iterations; but we cannot use this directly, because n depends on h .

We therefore analyse the series of substitutions taking place in the weakest liberal precondition for the discrete accumulation:

$$\begin{aligned} & x, \quad x - x \cdot h, \quad x - x \cdot h - (x - x \cdot h) \cdot h, \\ & (x - x \cdot h - (x - x \cdot h) \cdot h) - (x - x \cdot h - (x - x \cdot h) \cdot h) \cdot h, \quad \dots \end{aligned}$$

After taking common factors and performing series simplification, we obtain an accumulation function $\Phi(n \cdot h) = x(1 - h)^n$. Next we solve the h from the

accumulation guard $1 \cdot (1 - h)^n = 0.01$ yielding $h = 1 - e^{\frac{1}{n} \ln 0.01}$. Replacing the h in expression for c and simplifying it gives $c = n(1 - e^{\frac{1}{n} \ln 0.01})$. This expresses the time consumed by the lowering of the gate in proportion to the iteration accuracy n . Taking the limit of the Taylor polynomial of this expression when n approaches infinity gives an approximation for the total time, $-\ln 0.01 \approx 4.61$.

4.2 With continuous evolution

The same problem is expressed with continuous evolution by

$$x, c := 1, 0; \quad x > 0.01 : \dot{x}, \dot{c} = -x, 1$$

The total time is analysed as follows:

$$\begin{aligned} & \text{wlp} (x, c := 1, 0; \quad x > 0.01 : \dot{x}, \dot{c} = -x, 1, \quad x \leq 0.01 \wedge r = c) \\ &= \{ \text{definitions, instantiation of } \Phi_x(t) = xe^{-t} \text{ and } \Phi_c(t) = c + t \} \\ & \quad \forall \tau \geq 0 \cdot e^{-\tau} > 0.01 \vee r = \tau \vee \exists 0 \leq \delta < \tau \cdot e^{-\delta} \leq 0.01 \\ &= \{ \text{case analysis around } \tau = -\ln 0.01 \} \\ & \quad r = -\ln 0.01 \end{aligned}$$

This tells us directly that after $-\ln 0.01$ time units the gate becomes closed.

5 Hybrid Action System

A hybrid system starts from some initial state and alternates between continuous evolution and discrete changes of the variables in the system. The various circumstances, under which this alternation takes place, are studied more closely by Branicky [6].

5.1 Stutter free evolution

A differential action describes how the variables evolve when the guard is enabled. It also states that when the guard does not hold, the differential action has no effect on the variables. This is called stuttering. The equivalence below shows the decomposition of a differential action.

$$e : \dot{X} = F(X) \equiv e \rightarrow e : \dot{X} = F(X) \parallel \neg e \rightarrow \text{skip}$$

When thinking of how a hybrid system behaves, we are mainly interested in the stutter free behaviour. An infinite stuttering is seen as termination. Therefore, we introduce a notational short form for a stutter free differential action, which is $e : \dot{X} = F(X)$ with the meaning of $e \rightarrow e : \dot{X} = F(X)$. The stutter free differential action has the property that it is enabled only when the evolution guard holds, that is $\text{g}(e : \dot{X} = F(X)) = e$. Also, the enabledness set of a non-deterministic composition of stutter free differential actions is the disjunction of the evolution guards, that is,

$$\text{g}(e_1 : \dot{X} = F_1(X) \parallel \dots \parallel e_n : \dot{X} = F_n(X)) = e_1 \vee \dots \vee e_n$$

5.2 Hybrid alternation

The alternation between discrete and continuous behaviour is expressed by iteration of ordinary and (stutter free) differential actions. Assuming that the cause for the discrete behaviour D is calculated to be gD , and the stutter free evolution is known as DA , the alternation takes a normal form

$$\begin{array}{l} \text{do } gD \rightarrow D \\ \quad \parallel \neg gD \rightarrow DA \\ \text{od} \end{array} \quad (6)$$

The D in the model may be a non-deterministic composition of several discrete actions. Similarly, the stutter free evolution DA may be a non-deterministic composition of several stutter free differential actions.

For the form (6) of alternation above we introduce a short form

alt D **with** DA

This construct gives discrete changes priority over the continuous evolution. The continuous evolution may proceed only after all enabled discrete changes have been processed. However, once the continuous evolution is engaged it may continue without any interruption till the selected differential action terminates. After termination, the discrete changes may occur again. Thus, the model inserts discrete computations in between any continuous evolution described by a differential action. The alternation terminates, when it reaches a state in which no discrete changes and no continuous evolution is enabled, that is $\neg gD \wedge \neg gDA$. Hence, all the initial states from which the system terminates are given by $t(\text{alt } D \text{ with } DA)$.

The time in the model is assumed to progress during the continuous evolution, but not during the discrete changes. Therefore, if the discrete changes do not disable themselves, $\text{wp}(D, \neg gD) = \text{false}$, there is no continuous evolution nor a termination after the discrete changes become enabled. This corresponds to the system being *aborted*.

With the given form of alternation (6) we can identify a special class of action systems which have the desired hybrid behaviour.

Definition 2. Let X be local variables and Z be imported global variables. The actions in D and DA may refer to all of these variables X or Z . Any action system of the following form is called a hybrid action system.

$$\begin{array}{l} \mathcal{H} \hat{=} \parallel \text{var } X : T \\ \quad X := I; \\ \quad \text{alt } D \text{ with } DA \\ \parallel : Z \end{array}$$

When there are no discrete actions, i.e., $gD = \text{false}$, the **alt** part is dropped. Similarly, the **with** part is dropped, when there are no continuous evolutions, i.e., $gDA = \text{false}$.

A hybrid action system may speak also of the continuous evolution of imported global variables. The meaning of a hybrid action system is obtained by unfolding the hybrid alternation into an ordinary iteration (6). This is important for the definition of a non-trivial parallel composition.

6 Parallel Composition

Parallel composition is defined between action systems in [2]. Consider two action systems $\mathcal{A}$ and $\mathcal{B}$

$$\begin{array}{ll} \mathcal{A} \hat{=} \llbracket \text{var } X : T & \mathcal{B} \hat{=} \llbracket \text{var } Y : S \\ \quad X := I; & \quad Y := J; \\ \quad \text{do } A_1 \parallel \dots \parallel A_m \text{ od} & \quad \text{do } B_1 \parallel \dots \parallel B_n \text{ od} \\ \rrbracket : Z & \rrbracket : U \end{array}$$

where Z are imported global variables to $\mathcal{A}$ and U are imported global variables to $\mathcal{B}$, and the local variables X and Y are assumed to be disjoint. The parallel composition $\mathcal{A} \parallel \mathcal{B}$ is defined to be the action system

$$\begin{array}{l} \mathcal{A} \parallel \mathcal{B} = \llbracket \text{var } X, Y : T, S \\ \quad X, Y := I, J; \\ \quad \text{do } A_1 \parallel \dots \parallel A_m \parallel B_1 \parallel \dots \parallel B_n \text{ od} \\ \rrbracket : (Z \cup U) - (X \cup Y) \end{array}$$

The parallel composition combines the state spaces of the two action systems, merging the global variables and keeping the local variables distinct. Those imported global variables in $Z \cup U$, that are defined in either of the actions system, i.e. $(X \cup Y)$, are now locally declared.

The behaviour of the parallel composition depends on how the individual action systems, the *reactive components*, interact with each other via the global variables that they use. For instance, a reactive component does not terminate by itself. Termination is a global property of the composed action system [2].

Since the initialisations are well defined, and non-deterministic choice between actions is both associative and commutative, the parallel composition is also associative and commutative. Therefore, the meaning of several parallel composed action systems can be unfolded in any order without affecting the result.

6.1 Hybrid Action Systems

The parallel composition for action systems uses interleaving for parallelism. This works well for a discrete behaviour, but not for a continuous behaviour. Consider the previously presented gate example. Suppose, we try to compose another force lifting the gate with a constant velocity 0.5 m/s whenever the gate is at least half a meter away from being closed. Composing this force by interleaving the earlier non-linear closing force does not produce the desired result. By interleaving, the gate would non-deterministically either close normally, or it would start to open. In reality, we would expect that the gate stops half closed. The problem with the interleaving composition is that it does not really allow a simultaneous effect of the components, which is what we would like to have with the continuous behaviour. Motivated by this observation we explore an alternative way of composing continuous behaviour.

Linear Composition. In linear composition two partially defined functions are added on their common domain, whereas in the remaining domains the functions retain their original meaning. Linear composition thus assumes that a function is zero outside the domain where it is defined. The linear composition is formalised as follows. Let f be a partially defined function in the domain $\mathbf{dom} f$. Also, let g be a partially defined function in the domain $\mathbf{dom} g$. The linear composition of these functions is a set of partial functions

$$f(x) \oplus g(x) = \begin{cases} f(x) + g(x), & x \in \mathbf{dom} f \cap \mathbf{dom} g \\ f(x) + 0, & x \in \mathbf{dom} f \cap \overline{\mathbf{dom} g} \\ 0 + g(x), & x \in \overline{\mathbf{dom} f} \cap \mathbf{dom} g \end{cases} \quad (7)$$

The linear composition is both associative and commutative. It is defined in the whole range of $\mathbf{dom} f \cup \mathbf{dom} g$, whereas ordinary addition is defined only in the range $\mathbf{dom} f \cap \mathbf{dom} g$.

Obviously, the linear composition can be generalised for partially defined systems of differential equations. Let $\mathcal{F}(X)$ and $\mathcal{G}(X)$ be sets of vectors

$$\mathcal{F}(X) = \begin{cases} F_1(X), & X \in \mathbf{dom} F_1 \\ \vdots \\ F_m(X), & X \in \mathbf{dom} F_m \end{cases} \quad \mathcal{G}(X) = \begin{cases} G_1(X), & X \in \mathbf{dom} G_1 \\ \vdots \\ G_n(X), & X \in \mathbf{dom} G_n \end{cases}$$

where the domains $\mathbf{dom} F_1 \dots \mathbf{dom} F_m$ are disjoint, as well as the domains $\mathbf{dom} G_1 \dots \mathbf{dom} G_n$. A linearly composed system of differential equations

$$\dot{X} = \mathcal{F}(X) \oplus \mathcal{G}(X)$$

unfolds to a set of partially defined systems of differential equations

$$\left\{ \begin{array}{ll} \dot{X} = \mathcal{F}(X) + \mathcal{G}(X), & X \in \bigcup_{i=1}^m \mathbf{dom} F_i \cap \bigcup_{j=1}^n \mathbf{dom} G_j \\ \dot{X} = \mathcal{F}(X), & X \in \bigcup_{i=1}^m \mathbf{dom} F_i \cap \bigcap_{j=1}^n \overline{\mathbf{dom} G_j} \\ \dot{X} = \mathcal{G}(X), & X \in \bigcap_{i=1}^m \overline{\mathbf{dom} F_i} \cap \bigcup_{j=1}^n \mathbf{dom} G_j \end{array} \right.$$

Linear composition applies to stutter free differential actions. A stutter free differential action $e : \rightarrow \dot{X} = F(x)$ denotes a partially defined system of differential equations in the domain where e holds, that is $\{t \mid e[\phi(t)/X]\}$. A set of such systems with disjoint domains is a set of differential actions combined with non-deterministic choice.

Let DF be a composition $f_1 : \rightarrow \dot{X} = F_1(X) \parallel \dots \parallel f_m : \rightarrow \dot{X} = F_m(X)$, and DG be a composition $g_1 : \rightarrow \dot{X} = G_1(X) \parallel \dots \parallel g_n : \rightarrow \dot{X} = G_n(X)$. The DF and DG are analogous to $\dot{X} = \mathcal{F}(X)$ and $\dot{X} = \mathcal{G}(X)$. The linear composition of these functions presented above, is expressed in terms of actions as

$$\begin{array}{ll} \parallel i = 1..m, j = 1..n \cdot f_i \wedge g_j & : \rightarrow \dot{X} = F_i(X) + G_j(X) \\ \parallel i = 1..m \cdot f_i \wedge \neg gDG & : \rightarrow \dot{X} = F_i(X) \\ \parallel j = 1..n \cdot \neg gDF \wedge g_j & : \rightarrow \dot{X} = G_j(X) \end{array}$$

where gDF is $f_1 \vee \dots \vee f_m$ and gDG is $g_1 \vee \dots \vee g_n$.

Example Let X be a vector of variables x , y , and z . Let $F(X)$ be a vector of functions $(f(X), g(X), 0)$, and $G(X)$ be a vector of functions $(0, h(X), u(X))$. Consider the following linear composed system of differential equations where e and i are predicates over X used for describing the domains.

$$\dot{X} = F(X), \{X \mid e\} \oplus G(X), \{X \mid i\}$$

This unfolds to

$$\left\{ \begin{array}{ll} \dot{X} = (f(X), g(X) + h(X), u(X)), & \{X \mid e \wedge i\} \\ \dot{X} = (f(X), g(X), 0), & \{X \mid e \wedge \neg i\} \\ \dot{X} = (0, h(X), u(X)), & \{X \mid \neg e \wedge i\} \end{array} \right.$$

We use the differential actions to denote the same linear composition of systems of differential equations as follows.

$$e : \rightarrow \dot{X} = F(x) \oplus i : \rightarrow \dot{X} = G(X)$$

This unfolds into the following non-deterministic composition

$$\begin{array}{l} e \wedge i : \rightarrow \dot{x}, \dot{y}, \dot{z} = f(X), g(X) + h(X), u(X) \\ \parallel e \wedge \neg i : \rightarrow \dot{x}, \dot{y}, \dot{z} = f(X), g(X), 0 \\ \parallel \neg e \wedge i : \rightarrow \dot{x}, \dot{y}, \dot{z} = 0, h(X), u(X) \end{array}$$

6.2 Parallel composition of hybrid action systems

The separation of the continuous evolution and the discrete changes in a hybrid action system allows an extension of the parallel composition. We extend the parallel operator so, that it composes linearly the continuous evolution, but interleaves the discrete changes.

Let X and Y be disjoint local variables X , and Z and U be imported global variables. The ordinary actions H and stutter free differential actions DH may speak of variables X and Z , and similarly the ordinary actions K and stutter free differential actions DK may speak of variables Y and U . Consider the two hybrid action systems $\mathcal{H}$ and $\mathcal{K}$

$$\begin{array}{ll} \mathcal{H} \hat{=} \parallel \text{var } X : T & \mathcal{K} \hat{=} \parallel \text{var } Y : S \\ \quad X := I; & \quad Y := J; \\ \quad \text{alt } H \text{ with } DH & \quad \text{alt } K \text{ with } DK \\ \parallel : Z & \parallel : U \end{array}$$

where X and Y are disjoint. The parallel composition of these hybrid action systems has the following meaning.

$$\begin{array}{l} \mathcal{H} \parallel \mathcal{K} = \parallel \text{var } X, Y : T, S \\ \quad X, Y := I, J; \\ \quad \text{alt } H \parallel K \text{ with } DH \oplus DK \\ \parallel : Z \cup U - (X \cup Y) \end{array}$$

The meaning is obtained by first unfolding the linear composition for the continuous evolution and then by unfolding the hybrid alternation as before.

Since both interleaving composition and linear composition are associative and commutative, this extended parallel composition is also associative and commutative. This means that the composition of several hybrid action systems can be unfolded in any order without affecting the result.

A very simple example of parallel composition occurs, when we take an autonomous system, given by:

$$\begin{aligned} \mathcal{S} \hat{=} & \llbracket \text{var } X^* : T \\ & X := I; \\ & \text{with } true : \dot{X} = F(X) \\ & \rrbracket : \end{aligned}$$

and compose it with a control:

$$\begin{aligned} \mathcal{C} \hat{=} & \llbracket \text{var } V : S \\ & V := J; \\ & \text{alt } gK \rightarrow K \\ & \text{with } e : \dot{X} = U(X, V) \\ & \rrbracket : X \end{aligned}$$

The parallel composition gives:

$$\begin{aligned} \mathcal{S} \parallel \mathcal{C} = & \llbracket \text{var } X^*, V : T, S \\ & X, V := I, J; \\ & \text{alt } gK \rightarrow K \\ & \text{with } true : \dot{X} = F(X) \oplus e : \dot{X} = U(X, V) \\ & \rrbracket : \end{aligned}$$

which during evolutions satisfies $\dot{X} = F(X) + U(X, V)$. Control has been added.

7 A Water Tank Example

Parallel composition is illustrated by a water tank example. The water tank is a leaking container that is being filled, see Figure 2. Both leak and filling are continuous processes within the physical limitations of the tank. In this example, the task is to model these two different continuous processes as separate components and to obtain a model for the composed behaviour.

At the beginning, the container is empty. It starts leaking as soon as the water level denoted by y is at or above the lower point of an outlet. This threshold is denoted by min . Filling starts right from the beginning, and stops when the water level reaches the lower point of the inlet. This water level is denoted by max . The water flows in at rate $1 \text{ } l/s$ and it flows out at rate $\frac{1}{2} \text{ } l/s$.

7.1 Leaking

The water level y is a global variable, so that the filling process can share it. The hybrid system for the leaking consists only of an evolution describing the water flow from the tank whenever the water level is at or above min .

$$\begin{aligned} Leak \hat{=} & \llbracket \text{var } y^* : \mathbb{R} \\ & y := 0; \\ & \text{with } min \leq y : \dot{y} = -\frac{1}{2} \\ & \rrbracket : \end{aligned}$$

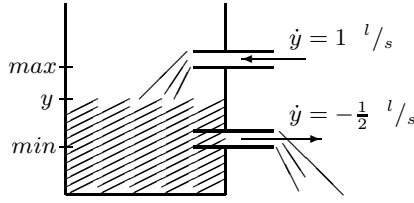


Fig. 2. The water tank with threshold water levels (min and max) and flow rates.

7.2 Filling

We use the global variable y from the previous action system to describe the water level. A global variable $fill$ records whether the tank is being filled or not.

The hybrid filling system alternates between a discrete decision to stop the filling when the water level hits the max threshold and a continuous evolution describing the water flow into the tank during the filling.

$$\begin{aligned} \mathcal{F}ill \hat{=} & \llbracket \text{var } fill^* : \mathcal{B} \\ & fill := true; \\ & \text{alt } fill \wedge y \geq max \rightarrow fill := false \\ & \text{with } fill \wedge y < max : \dot{y} = 1 \\ & \rrbracket : y \end{aligned}$$

The difference between this and a continuous evolution $\text{with } y < max : \dot{y} = 1$ is that the hybrid action system above fills the tank only once.

7.3 Tank

The behaviour for the water tank is obtained through a parallel composition of the models for filling and leaking.

$$\mathcal{T}ank \hat{=} \mathcal{F}ill \parallel \mathcal{L}eak$$

Unfolding the meaning of the water tank into a hybrid action system we get

$$\begin{aligned} \mathcal{T}ank = & \llbracket \text{var } fill, y^* : \mathcal{B}, \mathcal{R} \\ & fill, y := true, 0; \\ & \text{alt } fill \wedge y \geq max \rightarrow fill := false \\ & \text{with } fill \wedge y < max : \dot{y} = 1 \oplus min \leq y : \dot{y} = -\frac{1}{2} \\ & \rrbracket : \end{aligned}$$

Unfolding the linear composition yields

$$\begin{aligned} \mathcal{T}ank = & \llbracket \text{var } fill, y^* : \mathcal{B}, \mathcal{R} \\ & fill, y := true, 0; \\ & \text{alt } fill \wedge y \geq max & \rightarrow fill := false \\ & \text{with } fill \wedge y < min & : \dot{y} = 1 \\ & \parallel fill \wedge min \leq y < max & : \dot{y} = \frac{1}{2} \\ & \parallel (\neg fill \wedge min \leq y) \vee y \geq max : \dot{y} = -\frac{1}{2} \\ & \rrbracket : \end{aligned}$$

Finally, we translate the hybrid action system above into an ordinary action system. During the translation we use the properties for differential actions (Section 3.6) to simplify the result. The behaviour of the water tank is given by the action system

$$\begin{aligned}
 \mathcal{T}ank &= \llbracket \text{var } fill, y^* : \mathcal{B}, \mathcal{R} \\
 &\quad fill, y := true, 0; \\
 &\quad \text{do } fill \wedge y \geq max \quad \rightarrow fill := false \quad (1) \\
 &\quad \parallel fill \wedge y < min \quad \rightarrow y < min : \dot{y} = 1 \quad (2) \\
 &\quad \parallel fill \wedge min \leq y < max \rightarrow min \leq y < max : \dot{y} = \frac{1}{2} \quad (3) \\
 &\quad \parallel \neg fill \wedge min \leq y \quad \rightarrow min \leq y : \dot{y} = -\frac{1}{2} \quad (4) \\
 &\quad \text{od} \\
 &\rrbracket :
 \end{aligned}$$

When we analyse the behaviour of this action system, we notice that after initialisation only action (2) is enabled. This means that the tank first fills up to the *min* level at full rate. Then the action (3) becomes enabled; this action fills the tank up to the *max* level with half filling rate. When the tank is full action (1) becomes enabled; terminating filling. Immediately, action (4) becomes enabled letting the tank leak until the water goes below the *min* level. At this point, there are no more actions enabled, so the system terminates.

A reusable filling system might be modelled by adding a control that resets *fill*.

7.4 A step towards HyTech

The action system for the water tank above can also be translated to the language of HyTech. The translation is merely a representation of the behaviour described above. As a first step we rewrite the verbal presentation into a hybrid automaton yielding Figure 3. In this automaton each state represents a differential action in the system. The boolean variable *fill* is implicitly encoded in the names of the states. The discrete actions are seen as transition actions in between the states, and hence the discrete action (1) vanishes from the resulting automaton.

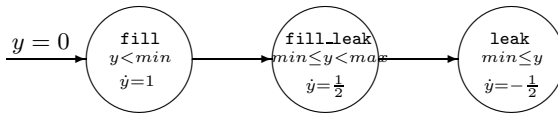


Fig. 3. A hybrid automaton describing the behaviour of the water tank.

The automaton in Figure 3 is a linear hybrid automaton, and therefore it can be written in the language of HyTech. The following is the HyTech input for the water tank automaton.

```

-- a water tank
var y : analog; -- the water level

automaton tank
syncclabs;;
initially fill & y=0; -- initialization
loc fill: while y<min wait {y'=1} -- action (2)
    when True goto fill_leak;
loc fill_leak: while min<=y & y<max wait {y'=1/2} -- action (3)
    when True goto leak;
loc leak: while min<=y wait {y'=-1/2} -- action (1),(4)
end -- tank

```

This idea of translating a hybrid action system into HyTech works well, as long as the continuous evolutions are described by differential equations with constant right-hand side (Not to be confused with linear differential equations, where the right hand side is a linear expression in the variables). This limitation comes from the theoretical model of linear hybrid automata, which is used as a basis in the tool. For the non-linear evolutions a linearisation must be performed before they can be expressed in the language of HyTech.

8 Conclusions

In this paper, we presented differential equations as primitive actions. Its weakest liberal precondition semantics was derived from the one for a discrete accumulation that approximates the continuous evolution. Differential actions provide a compact way of describing and analysing hybrid behaviours fusing standard mathematical analysis with discrete mathematics in the framework of weakest liberal preconditions as seen in the gate example.

We also identified a class of hybrid action systems. They alternate between discrete actions and evolutions as typical for hybrid systems. Since the hybrid action systems are based on the use of differential actions where the time is implicitly present, we avoid the complication of a global time variable and an associated tick action.

Parallel composition of action systems is extended to cover hybrid action systems. The extension is based on the mathematical properties of a linear composition. Since the use of hybrid action systems and the extended parallel composition does not impose any further constraints, all the existing results for action systems are directly usable. Moreover, the parallel composition between hybrid action systems and ordinary action systems remain meaningful with the extended parallel operator.

The analysis of the hybrid behaviour in a hybrid action system is obtained by translating it into an ordinary action system. As illustrated with the water tank example, the translation is mechanical and based on the given general model. An existing tool, HyTech, can well be used for detailed analysis of a hybrid

action system where the evolutions are described by differential equations with constant right-hand side or by approximating, cf. Theorem 2. This possibility was also illustrated with the water tank example.

These results are a foundation for further work. In particular, we shall investigate translations to HyTech of simple action systems. Furthermore, there is much remaining work in transferring results from control theory to the framework. Here, however, we consider it most fruitful to link this with specific classes of applications. A starting point could be a reformulation of Branicky's classification.

A very interesting idea is to study a variant of parallel composition, where the interaction between evolutions is through the differentiation variable. An example would be relative motion, where a person is walking with a speed $\frac{dp}{dx} = 1$ relative to a conveyor belt, satisfying $\dot{x} = f(x)$. Here parallel composition would be conjunction of the two equations. This is already included in the current definition of parallel composition for a single evolution; but what will happen during transitions?

Acknowledgments

We thank Ralph Back, Hans Rischel, Emil Sekerinski, Kaisa Sere, and the referees for their reading and comments for the earlier versions of this paper. This research was partially supported by the *Saastamoisen Säätiö* foundation. It has also been supported by the European Commission under Grant EC-US 043 Hybrid NSF. We would also like to thank the organisers of HS'97 for support.

References

1. R. Alur, C. Couroubetis, T. A. Henzinger, and Pei-Hsin Ho. Hybrid Automata: An Algorithmic Approach to the Specification and Verification of Hybrid Systems. In R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel (eds.) *Hybrid Systems*, LNCS 736, Springer-Verlag, 1993.
2. R. J. R. Back. *Refinement calculus, part II: Parallel and reactive programs*. In J. W. de Bakker, W.-P. de Roever, and G. Rozenberg, (eds.), *Stepwise Refinement of Distributed Systems: Models, Formalisms, Correctness. Proceedings. 1989*, LNCS 430, Springer-Verlag, 1990.
3. R. J. R. Back and R. Kurki-Suonio. Decentralization of process nets with centralized control. In *Proc. of the 2nd ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing*, pp. 131–142, 1983.
4. R. J. R. Back and K. Sere. Stepwise refinement of action systems. *Structured Programming*, **12**, pp. 17–30, 1991.
5. M. Bonsangue and J. N. Kok. Semantics, orderings and recursion in the weakest precondition calculus. In J. W. de Bakker, G. Rozenberg, W. P. de Roever (eds.) *Proceedings Rex Workshop on Semantics: Foundations and Applications*, LNCS 666, pp. 91–110, Springer-Verlag, 1993.
6. M. S. Branicky. *Studies in Hybrid Systems: Modeling, Analysis, and Control*. PhD Thesis. Massachusetts Institute of Technology, EECS Dept, 1995.
7. M. S. Branicky. General hybrid dynamical systems: modeling, analysis, and control. In R. Alur, T. A. Henzinger, and E. D. Sontag, editors, *Hybrid Systems III*, LNCS 1066, p. 186–200, Springer-Verlag, 1996.

8. M. Broy and G. Nelson. Adding Fair Choice to Dijkstra's Calculus. *ACM Transactions on Prog. Languages and Systems*, **16**,3, pp. 924-938, 1994.
9. M. Butler, E. Sekerinski, and K. Sere. An Action System Approach to the Steam Boiler Problem. In J-R. Abrial, E. Borger, H. Langmaack (eds.) *Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control*, LNCS 1165, Springer-Verlag, 1996.
10. K. Chandy, and J. Misra. *Parallel Program Design: A Foundation*. Addison-Wesley, 1988.
11. K. M. Chandy. Reasoning about continuous systems. *Science of Computer Programming*, **14**, pp. 117-132, 1990.
12. E. W. Dijkstra. *A Discipline of Programming*. Prentice-Hall International, 1976.
13. C.J. Fidge, A.J.Wellings. *An action-based formal model for concurrent, real-time systems*. Technical Report 95-1, SVR Center, January 1995.
14. He Jifeng. From CSP to Hybrid Systems. In *A Classical Mind, Essays in Honour of C.A.R. Hoare*, pp. 171 - 189, Prentice Hall, 1994.
15. T. A. Henzinger and P.-H. Ho. HYTECH: The Cornell hybrid technology tool. In P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, editors, *Hybrid Systems II*, LNCS 999, pp. 265-293, Springer-Verlag, 1995.
16. T. A. Henzinger, P-H. Ho, and H. Wong-Toi. *A User Guide to HyTech*. <http://www-cad.eecs.berkeley.edu/~tah/Publications>.
17. T. A. Henzinger and H. Wong-Toi. Linear phase-portrait approximations for non-linear hybrid systems. In R. Alur, T. A. Henzinger, and E. D. Sontag, editors, *Hybrid Systems III*, LNCS 1066, pp. 377-388, 1996.
18. Y. Kesten, Z. Manna, and A. Pnuelli. Verifying Clocked Transition Systems. In R. Alur, T. A. Henzinger, and E. D. Sontag (eds.) *Hybrid Systems III*, LNCS 1066, pp. 13-40, Springer-Verlag, 1996.
19. R. Kurki-Suonio. Stepwise Design of Real Time Systems. *IEEE Transactions on Software Engineering*, **19**, 1, pp. 56-69, 1993.
20. N. Lynch, R. Segala, F. Vaandrager, and H. B. Weinberg. Hybrid I/O automata. In R. Alur, T. A. Henzinger, and E. D. Sontag, editors, *Hybrid Systems III*, LNCS 1066, pp. 496-510, 1996.
21. M. Sintzoff. Abstract verification of structured dynamical systems. In R. Alur, T. A. Henzinger, and E. D. Sontag, editors, *Hybrid Systems III*, LNCS 1066, pp. 126-137, Springer-Verlag, 1996.
22. Xu Quiwen. Semantics and verification of extended phase transition systems in duration calculus. In O. Maler, editor, *Workshop on Hybrid and Real-Time Systems*, LNCS 1201, pp. 301-315, 1997.
23. Chaochen Zhou, Wang Ji, and A. P. Ravn. A formal description of hybrid systems. In R. Alur, T. Henzinger, and E. Sontag, editors, *Hybrid Systems III*, LNCS 1066, pp. 511-530. Springer-Verlag, 1996.

Towards Using Hybrid Automata for the Mission Planning of Unmanned Aerial Vehicles

Conrado W. Seibel¹, Jean-Marie Farines², and José E. R. Cury²

¹ Gyron Sistemas Autônomos Ltda.

Rua Lauro Linhares, 589

88036-002 Florianópolis (SC), Brazil

tel: +55 48 333 1591, fax: +55 48 333 3745

e-mail: conrado@gyron.com.br

² Laboratório de Controle e Microinformática

Universidade Federal de Santa Catarina

88040-900 Florianópolis (SC), Brazil

tel: +55 48 331 9202, fax: +55 48 331 9934

e-mail: {[farines](mailto:farines@lcmi.ufsc.br), [cury](mailto:cury@lcmi.ufsc.br)}@lcmi.ufsc.br

Abstract. This paper describes a methodology for the design of flight plans for rotary-wing unmanned aerial vehicles based on formal verification. The methodology uses linear hybrid automata to model the aircraft which will be used to perform a given mission, the flight plan that will be executed by that aircraft, the region where the flight will be performed and the meteorological conditions expected at the time the flight will be performed. The resulting model can be formally verified with respect to previously established safety and timeliness requirements, like not running out of fuel or keeping minimum distances from ground during all phases of the mission. The result of this verification can be used to instantiate values in a parameterized flight plan or to assist an operator in incrementally constructing a flight plan whose feasibility can be guaranteed in advance. The methodology is being embedded in a graphical flight plan editor which greatly reduces the time needed to plan a mission and increases the safety of the aircraft's operation.

1 Introduction

Unmanned Aerial Vehicles (UAVs) have been successfully used for reconnaissance purposes by military forces for over thirty years [Uni93]. More recently, the advantages of UAVs (low operational cost, no risk for human lives) have attracted the attention of civilian users and applications ranging from law enforcement and ecological surveillance [Kan94] to high-altitude scientific research [NAS93] have been identified.

Aircraft losses caused by human operator errors have traditionally been the biggest problem associated with UAVs [Ful96], having even determined the cancellation of some promising UAV programs. Running out of fuel during a mission is the second most common cause of UAV losses, highlighting the complexity of

balancing efficient use of the aircraft in pursuit of the mission objectives with conservative, safe operation.

Formal planning and verification of flight plans for UAVs can greatly increase the safety of operation of these aircraft, and reduce their operational cost by diminishing the workload imposed on the operator and therefore his/her training requirements.

One of the ways to tackle the task of formally planning a mission for an unmanned aircraft is using *hybrid automata*, a well suited approach for modeling and verifying hybrid systems.

In this paper we describe how hybrid automata can be used to model a rotary-wing unmanned aerial vehicle and its operating environment. This model can be verified with respect to safety criteria, like not running out of fuel, and can also be used to perform parametric analysis, allowing the instantiation of symbolic parameters in flight plans for specific missions.

Section 2 introduces aspects of rotary-wing unmanned aircraft relevant to this paper and describes the problem of generating flight plans for this class of aircraft.

Section 3 is a short review of modeling and verification using hybrid automata.

The use of hybrid automata to model a rotary-wing unmanned aircraft and its operating environment is presented in section 4.

In section 5 we show how the model developed can be used to *verify* a flight plan, i.e., to test if it is feasible, to *instantiate* a parameterized flight plan, i.e., to determine the boundaries of parameters in the flight plan so that it is feasible and to *incrementally construct* a flight plan whose feasibility can be guaranteed beforehand.

Finally, conclusions are drawn in section 6.

2 Mission Planning for Unmanned Aircraft

2.1 Definitions

We use the term *mission* to describe the operation of an aircraft in a given region, during a certain period of time while pursuing a specific objective.

The ordered set of movements executed by the aircraft during a mission is defined in an associated *flight plan*.

A flight plan can be decomposed in its component *phases*. Each phase is described either by the coordinates of a pair of *way-points* and by the speed at which the aircraft is to fly between these way-points or by its duration, an initial way-point and the speed of the aircraft. We consider a phase as completed when the second way-point is reached by the aircraft or, in the case of a specified duration, when the associated amount of time has elapsed.

For the purpose of this paper, we introduce the concepts of a *parameterized flight plan*, a plan with at least one symbolic parameter, and *flight plan instantiation*, the process by which symbolic parameters in a parameterized flight plan are instantiated to values for which specified requirements are satisfied.

2.2 The mission planning problem

Mission planning aims at producing a flight plan which allows the objectives of the mission to be pursued while complying with a set of mandatory and optional safety and timeliness *requirements*.

We consider as *internal conditions* to be taken into account in the design of the flight plan the resources and characteristics of the aircraft used for the mission. *External conditions* are defined as the operational context, dictated by the region where the flight is being performed and by the meteorological conditions expected at the time of the flight.

Internal conditions Each aircraft has a distinctive set of *resources*, which are used by the operator to pursue the mission objective. Resources have to be considered when planning a mission. They include the speed at which the aircraft can be operated, the fuel that can be carried on board, and electric energy which can be generated and/or stored on board the aircraft.

Also, an aircraft is characterized by the specific fuel consumption at each speed, fuel tank capacity and how much electric energy is generated, stored and managed on board.

External conditions To define the operational context of an unmanned aircraft we have to consider the prevailing *meteorological conditions* at the time the flight is performed. The aircraft's dynamics are affected by the speed and direction of the wind. This influence is most evident in small UAVs, which are often under-powered.

A flight plan also has to consider the *region* where the flight will be performed, i.e., the elevations of the terrain and the existence of “no-fly” zones, zones where the operation of the aircraft is not allowed at all or during certain periods of time.

The region where the flight is performed also influences the propagation of the radio waves used for communication between the aircraft and its ground control station. Particularly, the terrain profile can generate additional conditions when using line-of-sight communication links.

Safety and timeliness requirements A flight plan has to take a set of *safety requirements* into account. Mandatory safety requirements, like not running out of fuel, have to be satisfied to avoid the loss of the aircraft. Optional safety requirements, like keeping minimum distances from the ground during the flight, increase the safety of the operation.

Certain missions (mainly those involving search and interception) impose a time window on the opportunity to pursue an objective. Such missions have to satisfy not only safety requirements but also *timeliness requirements*.

3 Hybrid Automata: a short review

The characteristics and the dynamics of an unmanned aerial vehicle can be modeled by *hybrid automata*.

A hybrid automaton, as defined in [ACHH93,NOSY93,ACH⁺94], is constructed by the generalization of a finite-state automaton, equipped with a set of continuous variables. A hybrid automaton is able to model discrete events and continuous activities, governed by a set of differential equations.

Hybrid automata are described by a finite set of real-valued variables X and by a labeled multi-graph (V, E) . The standard notation $\dot{X}$ is used to denote the first-order derivatives of X . The edges E represent the discrete events and are labeled with restrictions on the values of X *before* and *after* the execution of the corresponding actions. The vertices V represent the continuous activities and are labeled with restrictions on X and $\dot{X}$ *during* the corresponding activity. Therefore, the state of a hybrid automaton is modified by discrete events and by the passing of time.

Formally, a hybrid automaton $H = (X, V, E, syn, act, inv)$ is composed of:

Variables A finite set $X = (x_1, x_2, \dots, x_n)$ of real-valued *variables*. The size n of X is called the *dimension* of H . A *valuation* of H is a point $s = (x_1 = a_1, x_2 = a_2, \dots, x_n = a_n)$ in the n -dimensional real space R^n and represents the state of the continuous variables of H . We will use S to describe the set of all possible valuations of H .

Locations A finite set V of vertices called *control locations*. A *state* of the automaton H is a pair (v, s) , where $v \in V$ is a control location and $s \in R^n$ is a valuation. The term *region* is used to denote a set of states. The valuations *associated with* a location v in a region W are the valuations s such that $(v, s) \in W$. We will use Σ to denote the set of all possible states of the automaton H .

Transitions A finite set of edges E called *transitions*. A transition $e = (v, a, \mu, v')$ consists of a *source* location $v \in V$, a *target* location $v' \in V$, a *synchronization label* $a \in syn$ and a *transition relation* $\mu \subseteq S^2$. The transition e is *enabled* in a state (v, s) if for some valuation $s' \in S$, $(s, s') \in \mu$. The state (v', s') is called a *transition successor* of the state (v, s) .

Synchronization Labels A finite set syn of *synchronization labels*, used to define the parallel composition of two automata: if two automata share the label α , then each α -transition in one automaton is followed by an α -transition in the other automaton.

Activities A function act that assigns to each location $v \in V$ a set of *activities*. Each activity is a function from the nonnegative reals $R^{\geq 0}$ to S . The model requires activities to be *time-invariant*: for all locations $v \in V$, activities $f \in act(v)$, and non-negative reals $t \in R^{\geq 0}$, also $(f + t) \in act(v)$, where $(f + t)(t') = f(t + t')$ for all $t' \in R^{\geq 0}$.

Invariants A function inv that assigns to each location $v \in V$ an *invariant* $inv(v) \subseteq S$. The automaton can stay in location v only while the invariant is true, that is, some discrete transition must be taken before the invariant becomes false.

Example 1. As an example, consider the hybrid system graphically represented in figure 1 (adapted from [ACHH93]).

The system has one data variable x and two locations, l_0 and l_1 . It starts in location l_0 , where the value of x decreases at a constant rate of 1. The transition from l_0 to l_1 may be taken at any time after the value of x has fallen below 6, and it must be taken before the value of x falls below 5. When the transition is taken, the value of x is instantaneously decreased by 1. Once on location l_1 , the value of x increases at a rate between 1 and 2. The transition back to location l_0 is taken exactly when the value of x hits 10.

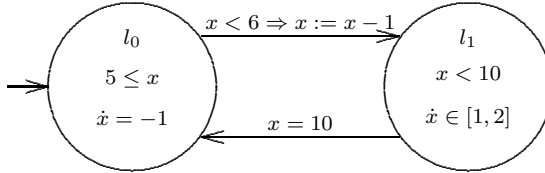


Fig. 1. Example of a hybrid automaton

A *hybrid system* is described by a collection of hybrid automata, one for each component of the system. The constituent automata operate in a concurrent and coordinated way, sharing a set of common variables X , and synchronizing on the common set $\text{syn}_1 \cap \text{syn}_2$ of synchronization labels.

Current verification techniques only allow the analysis of *linear* hybrid automata. A hybrid automaton is said to be linear if all its activities, invariants and transition relations can be described by linear expressions over the set of the automaton variables, X . This implies that for all locations $v \in V$, the activities $\text{act}(v)$ are defined by a set of differential equations of the form $\dot{x} = k_x$, one for each variable $x \in X$, where $k_x \in \mathbb{Z}$ is an integer constant.

There are various approximation techniques that allow a non-linear hybrid automaton to be converted in a linear one. For our purposes, the most useful of them is *rate conversion*, as described in [Ho95] and used in this paper.

4 Modeling Flight Plans with Hybrid Automata

As already mentioned, the problem of generating flight plans for UAVs can be tackled by *hybrid automata*. The position of the aircraft, the amount of fuel remaining on board the aircraft and the charge of the emergency accumulator are examples of continuous quantities while the transition from one phase of the flight plan to the next one is an example of a discrete transition of the same hybrid system.

Existing algorithms only allow the verification of linear hybrid automata. We therefore consider the following hypothesis, which allow us to describe flight plans for unmanned aircraft by a set of linear hybrid automata [SF97]:

- Position and velocity of the aircraft are described in a *tangent plane coordinate system*, a Cartesian coordinate system whose origin is located at the take-off/landing point. The error introduced by not taking into account the curvature of the Earth is negligible for the operational range of most UAVs (up to 500 km).
- The transitions from one phase of the flight plan to the next are much shorter than the duration of the phases themselves, allowing them to be ignored.
- The air-speed of the aircraft is constant during each phase of the flight. The speeds at which the aircraft can fly include best-range speed (the speed at which the *distance* traveled or *area* covered is maximized), endurance speed (the speed at which *airborne time* is maximized), maximum speed and, for rotary-wing aircraft, low speeds used for hover.
- The specific fuel consumption of the aircraft is not only a function of its speed but also a function of its mass, which decreases as the mission progresses and more fuel has been used. We use the *rate conversion* technique to linearize the specific fuel consumption of the aircraft.
- The specific fuel consumption is also a function of the altitude at which the aircraft flies. The influence can be neglected for low altitudes or be considered at the expense of the additional modeling steps required to linearize the automata.
- The terrain elevations, communications-covered regions and exclusion zones can be approximated with satisfactory resolution by a set of first-order inequalities.

Our approach to modeling an unmanned aircraft and its operating environment consists of:

- Modeling the dynamics of the aircraft under the influence of the prevailing (or expected) meteorological conditions with a hybrid automaton. We call this the $\mathcal{D}$ -automaton.
- Modeling the internal conditions (fuel consumption, electric charge of an emergency accumulator) with hybrid automata. We call this the $\mathcal{F}$ and $\mathcal{Q}$ -automata.
- Modeling the external conditions (terrain elevation, “no-fly” zones, radio-waves propagation) with inequalities over the tangent-plane coordinate system.

4.1 Aircraft dynamics

The aircraft’s dynamics, i.e., its position and velocity as a function of time, can be modeled by a linear hybrid automaton in which each control location corresponds to one of the phases of the flight plan.

The activities of each control location contain functions which describe the evolution of the aircraft’s position on a tangent plane coordinate system. Remember that, according to our previous assumption, the speed during each phase of the flight is constant.

The transitions connecting the different control locations are guarded by tests belonging to one of two categories:

- Tests on the aircraft's position. This type of test is used when the end of the n -th flight phase is defined by passing way-point $WP_{n+1} = (x_{n+1}, y_{n+1}, z_{n+1})$, specified by coordinates in the tangent plane coordinate system.
- Tests on a clock. This type of test is used when the duration of the n -th phase of the flight plan is specified, instead of its final way-point. The clock used has to be initialized when entering control location n .

Example 2. Consider a partial flight plan with only three phases:

1. Flight at best-range speed at a constant altitude of z meters, from way-point $WP_1 = (x_1, y_1, z)$ to way-point $WP_2 = (x_2, y_2, z)$.
2. Hover at $WP_2 = (x_2, y_2, z)$ for t_2 seconds.
3. Return to $WP_1 = (x_1, y_1, z)$ at maximum speed.

Figure 2 shows a linear hybrid automaton which models the dynamics of the partial flight plan just described.

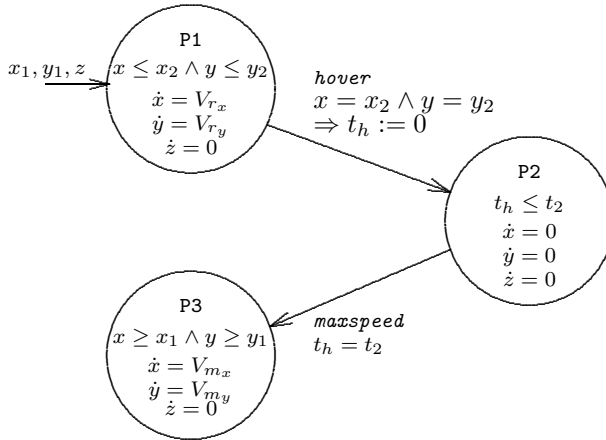


Fig. 2. A linear hybrid automaton used to model aircraft dynamics

The automaton starts at location P1. At this location, the aircraft's position in the tangent plane coordinate system is described by the differential equations $\dot{x} = V_{r_x}$ and $\dot{y} = V_{r_y}$. V_{r_x} and V_{r_y} are the north and east components of the aircraft's best-range speed V_r after considering the influence of wind:

$$V_{r_x} = V_r \cos \psi + V_w \cos \psi_w$$

$$V_{r_y} = V_r \sin \psi + V_w \sin \psi_w$$

where $\psi = \arctan((y_2 - y_1)/(x_2 - x_1))$ is the course that leads from WP_1 to WP_2 , ψ_w is the direction to where the wind blows and V_w is the wind's speed. It is important to note that the model accepts constant values for the wind's speed and direction as well as ranges of minimum/maximum values.

At $WP_2 = (x_2, y_2, z)$, the transition that leads to P2 is enabled and the invariant of P1 turns false. Firing the transition sets clock $t_h := 0$, which will be used to time the duration of the hover. The synchronization label *hover* is used to synchronize other automata used to model the aircraft.

The automaton remains at location P2 for t_2 seconds. Note that, during the hover, $\dot{x} = \dot{y} = 0$. After t_2 seconds have elapsed, the automaton progresses to location P3, which is used to model the return to WP_1 at maximum speed (*maxspeed*).

4.2 Specific fuel consumption

In general, the specific fuel consumption of a rotary-wing unmanned aircraft is a function of several variables and flight conditions which makes the resulting system a nonlinear hybrid one. In order to be analyzed, this system must be approximately modeled by a linear hybrid automaton. This is accomplished by the rate translation technique [Ho95], which replaces the nonlinear fuel consumption variable by a piecewise-linear variable that approximates the original one.

Let us consider, for example, a model for the specific fuel consumption (*sfc*) which reflects its dependency on the aircraft's mass and flight condition (hover, best range speed, endurance speed, maximum speed). Thus, in general, we have

$$sfc = \dot{f} = g_u(m)$$

where $m = m_e + f$ is the total aircraft's mass with m_e being the empty aircraft's mass, f the current fuel mass on board and $u \in \{\textit{hover}, \textit{bestrange}, \textit{endurance}, \textit{maxspeed}\}$ a parameter determining the flight condition. Under such conditions, g_u is obviously a non-positive function which decreases when m grows.

Let us assume g_u as being approximated by

$$g_u(m) = g_u(m_e) + h_u f$$

where $g_u(m_e)$ is the nominal specific fuel consumption for flight condition u , and $h_u f$ its increment due to the current amount of fuel f in the aircraft, h_u being a negative constant.

Considering a flight plan as the composition of several phases, each of them corresponding to a constant flight condition, we could model the fuel consumption of a rotary-wing unmanned aircraft by a nonlinear hybrid automaton such that location P_i models a phase with fuel consumption dynamics given by

$$sfc = k_{1_i} + k_{2_i} f$$

with k_{1_i} , k_{2_i} being non-positive constants corresponding to $g_u(m_e)$ and h_u for location i and flight condition u . Also, at each location P_i , f is constrained to be in the interval $[0, F]$ where F is the initial amount of fuel on board the aircraft.

A particular **NO-FUEL** location with dynamics $\dot{f} = 0$ is used to model the situation in which the aircraft runs out of fuel. Transitions in the automaton are either forced by synchronization labels representing changes in the flight conditions or by guarded transitions $f = 0$ which test the no-fuel condition and lead the automaton to the **NO-FUEL** location.

The resulting nonlinear hybrid automaton is bounded [Ho95] and can be translated into a linear hybrid automaton by approximating the nonlinear variable f by a piecewise-linear variable. A possible simple linear approximation is the one obtained by replacing a location P_i with a location satisfying

$$0 \leq f \leq F_i$$

and

$$\dot{f} \in [k_{1_i} + k_{2_i} F_i, k_{1_i}]$$

where F_i is calculated from a lower bound estimate of the fuel burned in the previous phases.

Figure 3 shows a linearized automaton which models the fuel consumption of an unmanned aircraft executing the partial flight plan shown in example 2.

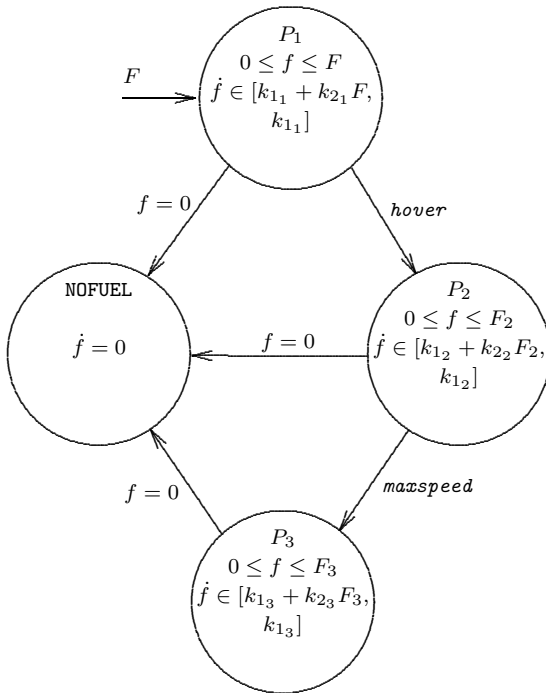


Fig. 3. A linearized hybrid automaton used to model the specific fuel consumption of a rotary-wing UAV

4.3 Emergency accumulator

The charge left on an emergency accumulator can be modeled by the automaton shown in figure 4. The location **CHARGE** is used to model the charging of the accumulator with a constant current I_{chg} . Assuming that the aircraft has some kind of energy management capabilities, discharging the accumulator is modeled by the locations **DISCHARGE1** (where the current drained from the accumulator is I_{dis}) and **DISCHARGE2** (where the current drained is $I_{min} < I_{dis}$). Two additional locations, **FULL** and **EMPTY**, are used to model the fact that the charge of an accumulator cannot be greater than its nominal capacity or lesser than zero.

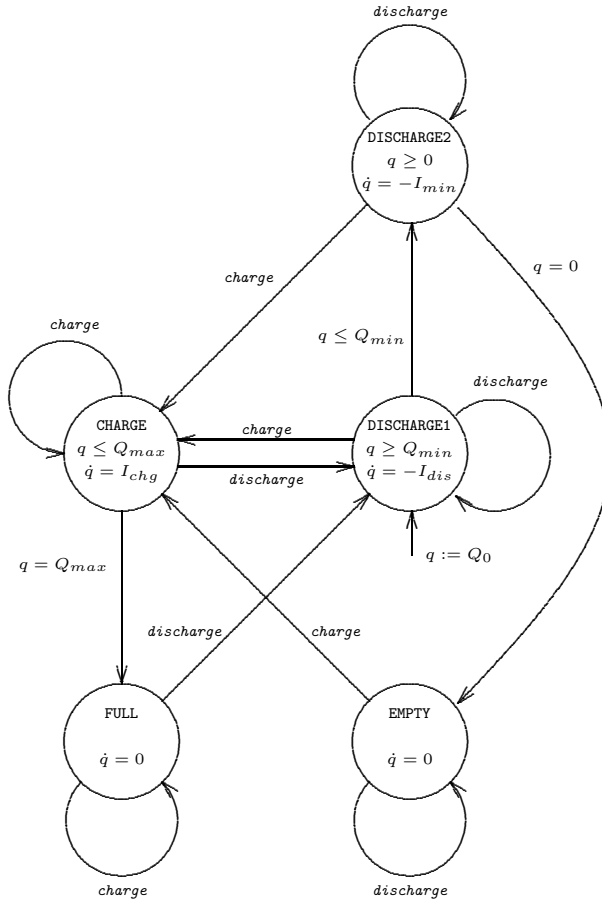


Fig. 4. A linear hybrid automaton used to model an emergency accumulator

Labels are used to synchronize this automaton with the rest of the model, so that the energy used during the initialization of the aircraft (when the engine

is not running and therefore the generator is not producing any energy) can be considered and failures of the generator/power conditioner can be modeled.

4.4 Communication coverage, terrain elevation and “no-fly” zones

We model communication coverage by a region, described by an inequality, where the signal is strong enough to guarantee the communication with the ground control station. This potentially non-linear region can then be approximated by the union of several linear regions.

Example 3. Assuming an omni-directional antenna and homogeneous propagation of radio-frequency waves in a given terrain, the coverage zone is described by a circle of radius R_{max} . Let us assume that the antenna is located at the takeoff/landing point, the origin of the tangent plane coordinate system. We therefore have that a strong enough signal in the region described by following inequality:

$$x^2 + y^2 \leq R_{max}^2$$

This non-linear region can be conservatively approximated by the union of several linear regions:

$$\begin{aligned} -x_1 &\leq x \leq +x_1 \wedge -y_1 \leq y \leq +y_1 \quad \vee \\ -x_2 &\leq x \leq +x_2 \wedge +y_1 \leq y \leq +y_2 \quad \vee \\ -x_2 &\leq x \leq +x_2 \wedge -y_2 \leq y \leq -y_1 \quad \vee \\ -x_3 &\leq x \leq +x_3 \wedge +y_2 \leq y \leq +y_3 \quad \vee \\ -x_3 &\leq x \leq +x_3 \wedge -y_3 \leq y \leq -y_2 \quad \vee \\ &\vdots \end{aligned}$$

Figure 5 illustrates graphically the approximation of $x^2 + y^2 \leq R_{max}^2$ by the set of regions presented above.

In a similar way, terrain elevation and “no-fly” zones are modeled by three-dimensional regions.

4.5 Meteorological conditions

The influence of wind has been already described when dealing with the aircraft dynamics (section 4.1).

5 Design of Flight Plans with Hybrid Automata

In the following we present different ways to design flight plans for unmanned aerial vehicles based on formal verification of hybrid automata.

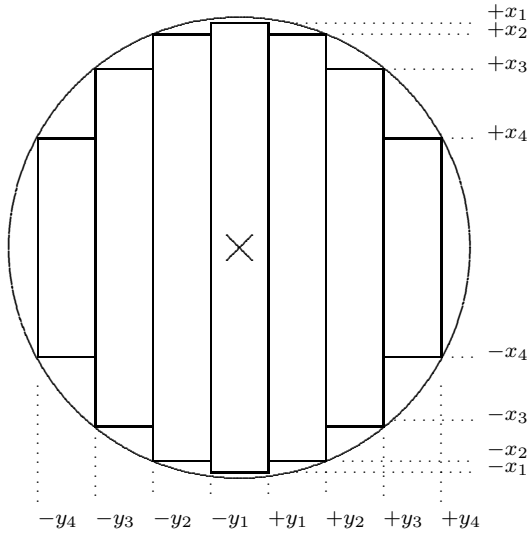


Fig. 5. Modeling of communications coverage

A methodology for the analysis of linear hybrid automata is presented in [ACH⁺94,AHH96,Yov93]. The methodology is based on predicate transformers for computing the step predecessors and step successors of a given set of states.

Using successive approximation, the methodology allows the computation of $post^*(W)$, the *reachable region* of a region W , i.e., the set of all states that are reachable from states in W . Conversely, it is possible to compute $pre^*(W)$, the *initial region* of W , the set of all states from which a state in W is reachable.

The more general problem of verifying if a hybrid automaton satisfies a requirement expressed in a formula of the real-time logic TCTL (timed computation tree logic) is also addressed in [ACH⁺94].

The hybrid automata modeling the mission of an unmanned aircraft can be formally verified with respect to safety and timeliness criteria. We call this *formal verification of the flight plan associated with the mission*. In this paper, we will only discuss the verification of safety requirements.

One of the ways of specifying safety criteria is to use a real-time temporal logic like TCTL. By using the operators *never*, *always* and *eventually*, it is possible to describe the desired behavior of the hybrid automaton being verified.

Another way of specifying safety criteria is by using the concept of *regions*, sets of states of the hybrid automaton being investigated. All the safety criteria that flight plans for UAVs have to satisfy can be expressed in terms of inclusion in a “good” region and intersection with a “bad” region.

We use regions to specify safety criteria for mission planning since three-dimensional regions on the tangent-plane coordinate system seem very natural to our problem.

Flight plans can be designed in three different ways:

- The feasibility of a proposed flight plan can be *verified*. The result of the verification process is either “flight plan is feasible” or “flight plan is not feasible”.
- Symbolic parameters in a parameterized flight plan can be *instantiated* so that the resulting flight plan is feasible. The result of the instantiation process is “flight plan will be feasible if ...”.
- Flight plans can be incrementally constructed by computing the safely reachable region as each phase is added to the partially constructed flight plan. The overall feasibility of a flight plan constructed in this manner is therefore guaranteed in advance.

5.1 Verification of flight plans

Using regions, the problem of verifying the feasibility of a flight plan can be summarized as follows:

1. Construct an automaton that describes the mission (the $\mathcal{M}$ -automaton). This is the parallel composition of the $\mathcal{D}$, $\mathcal{F}$ and $\mathcal{Q}$ -automata.
2. Starting with initial conditions at the take-off point (region I), compute the forward reachable region of the $\mathcal{M}$ -automaton.
3. Compute the intersection of the reachable region with the “bad” region (defined by the external conditions in the form of inequalities on the tangent plane coordinate system). If the intersection is not empty, the feasibility of the flight plan cannot be guaranteed.
4. Verify if the reachable region is contained within the “good” region. If not, the feasibility of the flight plan cannot be guaranteed.

Tools for the automatic analysis of hybrid systems, like HyTECH (described in [HHWT95, HH95]), can be used to perform the parallel composition of hybrid automata and to compute the reachable region of the resulting hybrid system. HyTECH is also able to manipulate regions, computing their intersection and verifying if a region is included within another one.

5.2 Instantiation of parameterized flight plans

The methodology for the analysis of hybrid systems presented in [ACH⁺94] is able to manipulate not only numerical quantities but also *symbolic parameters*.

We can use symbolic parameters to *instantiate* parameterized flight plans, i.e., determine for which values of the symbolic parameters the safety requirements are satisfied. Instantiation can be summarized as follows:

1. Construct an automaton that describes the mission (the $\mathcal{M}$ -automaton). This is the parallel composition of the $\mathcal{D}$, $\mathcal{F}$ and $\mathcal{Q}$ -automata.
2. Compute the backward reachable region of the $\mathcal{M}$ -automaton, starting at the overall “bad” region.

- 3. Intersect the result with the initial region I and instantiate all symbolic parameters. The values obtained at this step are those that will lead from the initial region to the “bad” region.
- 4. Complement the parameters obtained in the previous step. These are the values for which the mission is feasible.

5.3 Incremental construction of flight plans

Incrementally constructing a flight plan can be done by continuously computing and displaying the reachable region of the $\mathcal{M}$ -automaton over a map of the terrain where the flight will be performed, as shown in figure 6.

The consequences of adding or removing way-points to the flight plan are immediately reflected in the shape of the reachable region, assisting the operator in the determination of the next phase of the flight plan being constructed.

Further details of such a *graphical flight plan editor* with an underlying verification engine can be found in [SFCD98].

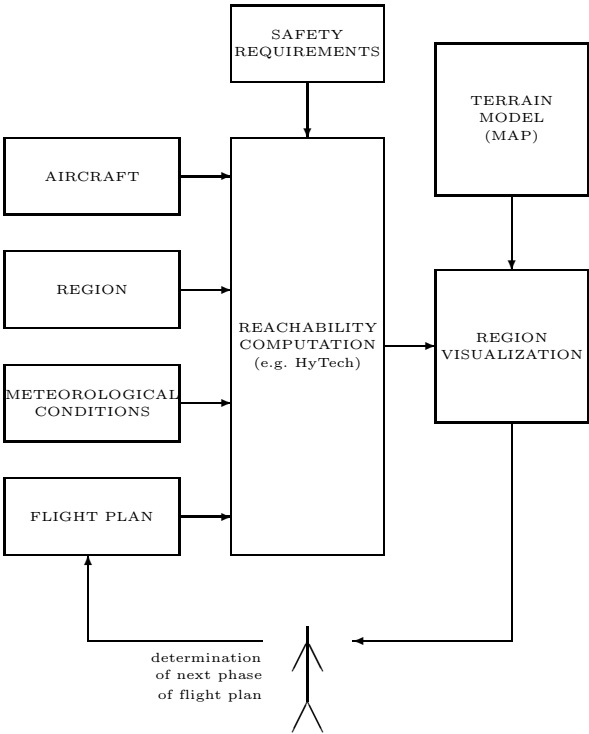


Fig. 6. Incremental construction of flight plans

5.4 Example of flight plan verification

Let us consider the sample mission shown in figure 7. The mission consists of over-flying a couple of islands off the coast of Santa Catarina in the southern part of Brazil while investigating outlawed fishing activities on the islands. Take-off/landing point coordinates are $27^\circ 40.9' \text{ S}$, $48^\circ 33.8' \text{ W}$. Coordinates of Island Francisca are $27^\circ 42.2' \text{ S}$, $48^\circ 33.9' \text{ W}$ and of Island do Largo $27^\circ 42.4' \text{ S}$, $48^\circ 35.6' \text{ W}$. Tangent plane coordinates are, respectively, $(0, 0)$, $(-2247, -118)$ and $(-3051, -3080)$ ¹. The aircraft will be flown at its best range speed, 15 m/s. Specific fuel consumption is less than 0.56 g/s under such conditions.

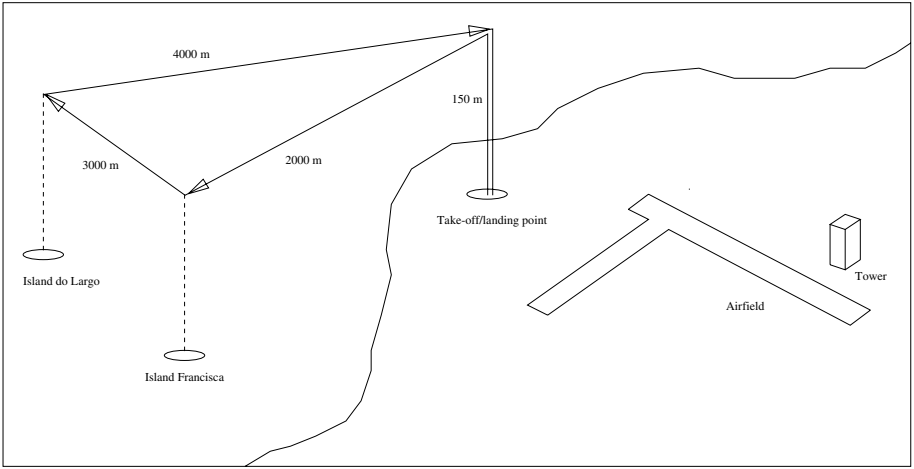


Fig. 7. Sample mission

We want to verify if an initial amount of fuel $F = 1000 \text{ g}$ is sufficient to accomplish the mission while avoiding the approach corridor of the nearby airfield, described (in tangent plane coordinates) by

$$-2/5y + 700 < x < -2/5y + 3700$$

Using HYTECH we find that the reachable region of the $\mathcal{M}$ -automaton is

$$\begin{aligned} post^*(I) = 0 = & x + 15t \wedge 4x = 75y \wedge 27f = x + 27000 \wedge \\ & 0 \geq x \wedge x + 2250 \geq 0 \\ \vee 0 = & x + 4t + 1650 \wedge 8y = 29x + 64290 \wedge 36f = 5x + 44250 \wedge \\ & 0 \geq x + 2250 \wedge x + 3045 \geq 0 \\ \vee 32t = & 3x + 20295 \wedge 63x = 64y + 285 \wedge 5x + 96f = 62175 \wedge \end{aligned}$$

¹ tangent plane coordinates are expressed in meters

$$\begin{aligned}
& x + 3045 \geq 0 \wedge 0 \geq x \\
& \vee x = 0 \wedge 64y + 285 = 0 \wedge 32f = 20725 \wedge 32t \geq 20295
\end{aligned}$$

We can easily verify from the above result that the aircraft uses $\frac{20725}{32}$ g of fuel (which is less than $F = 1000$ g) and that the intersection of the reachable region of the $\mathcal{M}$ -automaton and the “no-fly” zone defined by the approach corridor to the airfield is empty. We can therefore conclude that the proposed flight plan is feasible.

6 Conclusions and Further Work

Hybrid automata are well suited to model unmanned aircraft and their operating environment and the usage of regions is very natural for the verification of safety requirements.

We are generating flight plans for the HELIX UAV [Ltd92], shown in figure 8, with a prototype mission planning tool that implements the verification, instantiation and incremental construction of flight plans techniques described in this article. Our experience is that the use of this techniques greatly increases the confidence of the operator, even if working around some overflow problems of HyTECH very often requires manual intervention in the generated models.



Fig. 8. UAV for which a prototype mission planning tool is being used

In the near future we intend to use the methodology not only for mission planning (construction and verification of flight plans prior to their execution) but also for mission control (during the execution of the mission). In this situation, a model will be constructed and verified on-line periodically, providing an additional level of supervision when the aircraft is operated in an interactive mode.

References

- ACH⁺94. R. Alur, C. Courcoubetis, T. Henzinger, P. Ho, X. Nicollin, A. Olivero, and S. Yovine. The algorithmic analysis of hybrid systems. In *Proceedings of the 11th International Conference on Analysis and Optimization of Discrete Event Systems*, pages 331–351. Springer Verlag, 1994.
- ACHH93. R. Alur, C. Courcoubetis, T. Henzinger, and P. Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, editors, *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 209–229. Springer Verlag, 1993.
- AHH96. R. Alur, T. Henzinger, and P. Ho. Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering*, 22(3), March 1996.
- Ful96. David A. Fulghum. Outrider UAV tackles army, navy requirements. *Aviation Week & Space Technology*, July 22 1996.
- HH95. T. Henzinger and P. Ho. HYTECH: The Cornell HYbrid TECHnology tool. In *Proceedings of 1994 Workshop on Hybrid Systems and Autonomous Control*. Springer Verlag, 1995.
- HHWT95. T. Henzinger, P. Ho, and H. Wong-Toi. A user guide to HYTECH. In *TACAS 95: Tools and Algorithms for the Construction and Analysis of Systems*. Springer Verlag, 1995.
- Ho95. Pei-Hsin Ho. *Automatic Analysis of Hybrid Systems*. PhD thesis, Cornell University, Department of Computer Science, 1995.
- Kan94. S. Kandebo. CYPHER moves toward autonomous flight. *Aviation Week & Space Technology*, March 7 1994.
- Ltd92. Gyron Sistemas Autônomos Ltda. Projeto HELIX: Definição e descrição geral. Internal document, unpublished, 1992.
- NAS93. NASA. Extending the vision. *Unmanned Systems*, 11(1), 1993.
- NOSY93. X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. An approach to the description and analysis of hybrid systems. In R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, editors, *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 149–178. Springer Verlag, 1993.
- SF97. C. Seibel and J.M. Farines. Formal verification and dimensioning of flight plans for rotary-wing unmanned aerial vehicles. In *Proceedings of the 24th Annual Technical Symposium of the Association for Unmanned Vehicle Systems International*, Baltimore, MD, June 1997.
- SFCD98. C. Seibel, J.M. Farines, L. C. Corrêa, and G. C. Dallagnolo. Generating executable flight plans for unmanned aircraft: A formal mission planning tool. In *Proceedings of the 13th Bristol International Conference on R-PVs/UAVs*, Bristol, UK, March 1998.
- Uni93. J. Unitt. A bright future for RPVs after Vietnam? *Unmanned Systems*, 11(2), 1993.
- Yov93. S. Yovine. *Méthodes et Outils pour la Vérification Symbolique de Systèmes Temporisés*. PhD thesis, Institut National Polytechnique de Grenoble, 1993.

Optimal Controller Switching for Stochastic Systems

E. Skafidas¹, R.J. Evans², I.M.Y. Mareels², and A. Nerode^{3*}

¹ Hybrithms Corporation,
10632 NE 37th Circle, Bldg. 23
Kirkland, WA 98033, USA
`skafidas@hybrithms.com`

² Center for Sensor Signal and Information Processing,
Dept. of Electrical and Electronic Engineering,
University of Melbourne, Parkville, Vic 3052, Australia
`{robevans,iven}@ee.mu.oz.au`

³ Department of Mathematics,
Cornell University,
Ithaca, NY 14850, USA
`nerode@hybrid.cornell.edu`

Abstract. This paper presents a solution to certain problems in switched controller design for stochastic dynamical systems with a quadratic cost. The main result is a separation theorem for partial information systems. This result is then used to convert the partial information stochastic control problem to a complete information stochastic control problem. We also show that certainty equivalence does not hold. The optimal sequence of controllers can be determined via an appropriate solution to a dynamic programming problem.

1 Introduction

There are many real industrial control problems where the control action is determined by switching among a finite number of given control laws. Typical examples include an automobile transmission, where different gears need to be switched in order to obtain desirable performance (gain switching) [1,2], a thermostat where the plant heats up when the control is on and cools when the control is off, and steering an automobile where switching between appropriately chosen controls laws can make an otherwise uncontrollable vehicle controllable and allow the vehicle to be steered (locally) in any desired direction [3](pg 76).

In [4,5] we devised optimal strategies for switching between given controllers at fixed switching intervals for plants described by an uncertain differential equation. In this paper we consider optimal switching strategies for plants described by linear time-varying stochastic differential equations with a quadratic

* This work was supported by the Center for Sensor Signal and Information Processing the Australian Research Council and by US Army Research Office under the MURI grant “An Integrated Approach to Intelligent Systems”, grant DAAH04-96-1-0341.

cost functional and develop optimal switching strategies for both the full and partial information cases [6,7]. We consider separation and certainty equivalence for switched systems. Although these notions have been studied for many years [8] and their applicability to various classes of stochastic systems continues to be of great interest e.g. [9]. To our knowledge these important notions have not been investigated for linear switched controller and hybrid control systems. This paper presumes a background knowledge of linear stochastic control systems and Ito calculus. An excellent treatment of these subjects can be found in [10] and [11].

In this paper we show that the full information problem with quadratic cost the optimal switching sequence of controllers is given by a solution to a dynamic programming equation. We show that where the switching function depends on both time and state (state estimate) certainty equivalence does not hold. For the partial information case a separation theorem is established. This is then used to convert the partial information problem to a full information problem enabling the optimal controller switching sequence to be determined.

The organization of this paper is as follows. Section 2 considers full information switched controller systems. The main results of this section is to show equivalence between the optimal controller switching sequence and the solution to a dynamic programming equation. In this section it is also shown that for the class of systems considered in this paper certainty equivalence does not hold. In section 3 the partial information problem is considered. The main results of this section are given in terms of the existence of suitable solutions to a Ricatti differential equation of the Kalman Filtering type and appropriate solutions to a dynamic programming equation. In section 4 a simulation example is presented.

2 Optimal Controller Switching with Full Information

All random variable in this paper are defined on the probability space $(\Omega, \mathcal{A}, \mathcal{P})$. Let $w(\cdot)$ be a vector valued Wiener process with covariance matrix $Q(\cdot)$. Let $\mathcal{E}$ denote expectation with respect to the measure $\mathcal{P}$. Let $N \in \mathbf{N}^+$ be a given positive integer. The permissible switching times $\{t_0, t_1, t_2, \dots, t_N\}$ corresponding to an ordered discrete set where $(t_i \in \mathbf{R}^+)$.

Consider the following stochastic system whose dynamics on the interval $[0, t_N)$ are given by the stochastic differential equation

$$dx(t) = (A(t)x(t) + B_2(t)u(t))dt + B_1(t)dw(t); \quad x(0) = x_0 \quad (1)$$

where $x(t) \in \mathbf{R}^n$ denotes the state, $u(t) \in \mathbf{R}^h$ is the control and the functions $A(t)$, $B_1(t)$, $B_2(t)$, $C(t)$ are given piecewise continuous matrix functions of time of appropriate dimension.

Controller Switching with State Feedback [12,5,4,13]. Given a finite collection of controllers

$$u_1(t) = U_1(t, x(t)), \quad u_2(t) = U_2(t, x(t)), \quad \dots, \quad u_k(t) = U_k(t, x(t)) \quad (2)$$

where the control laws $U_1(\cdot, \cdot), U_2(\cdot, \cdot), \dots, U_k(\cdot, \cdot)$ are given continuous matrix functions. Let $I_j(\cdot)$ denote a state switching function which is permitted to switch between controllers at set switching times t_j , $\{j = 1, \dots, N\}$. The class of switching functions considered in this section is of the form $I_j(\cdot) : \{x(\cdot) \mid_0^{t_j}\} \rightarrow \{1, 2, \dots, k\}$, which map from the set of state measurements $\{x(\cdot) \mid_0^{t_j}\}$ to the set of symbols $\{1, 2, \dots, k\}$ which index the controllers (2). This class of switching functions defines a family of dynamic nonlinear state feedback controllers of the form:

$$\forall j \in \{0, 1, \dots, N-1\} \quad u(t) = U_{i_j}(t, x(t)) \quad \forall t \in [t_j, t_{(j+1)}) \quad \text{where } i_j = I_j(x(\cdot) \mid_0^{t_j}). \quad (3)$$

Hence the control strategy, which is a rule for switching from one controller to another, constructs a symbolic sequence $\{i_j\}_{j=0}^{N-1}$ from the set of state measurements $x(\cdot) \mid_0^{t_j}$. The set of all admissible controls achievable by controller switching (3) with the controllers (2) is denoted by $\mathcal{U}$.

Let $S(\cdot)$ be a given function which maps from R^n to R , let $N(t)$ and $M(t)$ be given matrices of time satisfying $N(t), M(t) \geq \rho I$, $\rho > 0 \quad \forall t \in [0, t_N]$. $X_f \geq 0$ is a given constant matrix which penalises the final state.

Introduce the following quadratic cost function,

$$W(t, x, u(t)) \triangleq \mathcal{E} \left[x'(t)N(t)x(t) + u'(t)M(t)u(t) \right] \quad (4)$$

and define

$$F_j^i(x_0, S(\cdot)) \triangleq \mathcal{E} [S(x(t_{j+1}))] + \int_{t_j}^{t_{j+1}} W(t, x, u_i(t)) dt. \quad (5)$$

Definition 1 (State Feedback Stochastic Control Problem). *The state feedback stochastic control problem with system (1) is said to have a solution via controller switching with the controllers (2) if the following conditions hold:*

- (i) *For any admissible control sequence, $u \in \mathcal{U}$, there exists a unique trajectory of system (1) on the interval $[0, t_N]$.*
- (ii) *The cost*

$$J(u)_{u \in \mathcal{U}} = \mathcal{E} \left(\int_0^{t_N} x'(t)N(t)x(t) + u'(t)M(t)u(t) dt + x'_{t_N} X_f x_{t_N} \right) \quad (6)$$

exists and is finite for some admissible control sequence, $u \in \mathcal{U}$.

Theorem 1. *The state feedback stochastic control problem (Definition (1)) for system (1) has a solution via controller switching with the state feedback controllers (2) if and only if the dynamic programming problem*

$$V_N(x_0) = \mathcal{E} [x'_{t_N} X_f x_{t_N}] ; \quad V_j(x_0) = \min_{i=1, \dots, k} \mathcal{E} [F_j^i(x_0, V_{j+1}(\cdot))] \quad (7)$$

has a solution for all $j = 0, 1, \dots, N-1$ and all $x_0 \in \mathbf{R}^n$.

Furthermore let $i_j(x_0)$ be an index such that the minimum in (7) is achieved for $i = i_j(x_0)$. Then the controller (2), (3) associated with the switching sequence $\{i_j\}_{j=0}^{N-1}$ where $i_j \triangleq i_j(x(t_j))$ solves the state feedback stochastic control problem.

Proof. Sufficiency. For any $j = 0, 1, \dots, N$ introduce the cost to go

$$V_j \triangleq \min_{u(t) \in \mathcal{U}} \mathcal{E} \left(\int_{t_j}^{t_N} x'(t)N(t)x(t) + u'(t)M(t)u(t)dt + x'_{t_N}X_fx_{t_N} \right). \quad (8)$$

Following [14,10] $V_j(\cdot)$ satisfies (7). Also it follows from (8), that if the stochastic control problem has a solution, then $V_0(x_0)$ exists and is finite.

Necessity. Equation (7) implies that for the controller associated with the switching sequence $\{i_j\}_{j=0}^{N-1}$ we have

$$\mathcal{E} \left(\int_0^{t_N} x'(t)N(t)x(t) + u'(t)M(t)u(t)dt + x'_{t_N}X_fx_{t_N} \right) \leq V_0(x(0)). \quad (9)$$

Since $V_0(x(0)) = J(u^*, x(0)) \leq J_{u \in \mathcal{U}}(u, x(0))$, the controller associated with the switching sequence, $\{i_j\}_{j=0}^{N-1}$, solves the state feedback stochastic control problem.

Suppose that the given state feedback controllers (2) are of the form

$$u_1(t) = K_1(t)x(t), \quad u_2(t) = K_2(t)x(t), \quad \dots, \quad u_k(t) = K_k(t)x(t) \quad (10)$$

where $K_i(t)$ are given matrices of time, then we have.

Definition 2 *Certainty Equivalence [10,6]. The system (1) is said to satisfy a certainty equivalence principle if the optimal admissible control law is independent of the statistics of the noise inputs.*

Corollary 1 *Consider system (1) and state feedback controllers (10). For this class of stochastic control problems certainty equivalence does not hold.*

Proof. Consider system (1) with the state feedback basic controllers (10). All that is required to be shown is that the controller switching sequence for the plant with no noise is different to that with noise.

The cost $J(u)$ for all admissible controllers can be rewritten as

$$J_{u \in \mathcal{U}}(u) = \mathcal{E} \left(x'_{t_N}X_fx_{t_N} + \int_0^{t_N} x'(t)(N(t) + K'_{i_t}R(t)K_{i_t})x(t)dt \right). \quad (11)$$

Expressing the state as

$$x(t) \triangleq \tilde{x}(t) + x^*(t) \quad (12)$$

with

$$\dot{\tilde{x}}(t) = (A + B_2K_{i_t})\tilde{x}(t), \quad \tilde{x}(0) = x(0) \quad (13)$$

$$dx^*(t) = (A + B_2K_{i_t})x^*(t)dt + dw(t) \quad x^*(0) = 0, \quad (14)$$

it follows that

$$\mathcal{E}x^*(t) = 0, \mathcal{E}\tilde{x}'(t)x^*(t) = \tilde{x}'(t)\mathcal{E}x^*(t) = 0 \quad \forall t. \quad (15)$$

Following standard dynamic programming we substitute of (14) and (15) into (11) and rewriting (11) as:

$$\begin{aligned} J_{u \in \mathcal{U}} = & \mathcal{E} \left(x'_{t_N} X_f x_{t_N} \right) + \mathcal{E} \left(\int_{t_{N-1}}^{t_N} \tilde{x}'(t) (N(t) + K'_{i_t} R(t) K_{i_t}) \tilde{x}(t) dt \right) \\ & + \mathcal{E} \left(\int_{t_{N-1}}^{t_N} x^{*'}(t) (N(t) + K'_{i_t} R(t) K_{i_t}) x^*(t) dt \right) \\ & + \mathcal{E} \left(\sum_{n=0}^{N-2} \int_{t_n}^{t_{n+1}} x'(t) (N(t) + K'_{i_t} R(t) K_{i_t}) x(t) dt \right). \end{aligned} \quad (16)$$

It can now be seen that in general the choice of gain matrix that optimizes the second and third terms in the above expression (16) will depend on both the state as well as the noise covariance and its contribution to the state over the interval $[t_{N-1}, t_N]$. This contradicts certainty equivalence.

Remark In switched controller systems the optimal controller switching sequence for linear feedback controllers depends on the input noise covariance. This result, also applies to zero-mean white Gaussian process noise inputs. This differs from optimal linear quadratic Gaussian control [7,6,15] where the optimal feedback gain is independent of the input noise covariance.

We illustrate that Corollary (1) above by considering the system (1) with parameters (29) in the simple scenario corresponding to only one switching time with switching occurring at $T = 0$. Let the system be defined over the interval $[0, 0.1]$ with $X_f = 0$. The process noise $w(t)$ is zero mean Gaussian with covariance $\mathcal{E}(w(t)w'(t)) = 1.0$. For this plant the cost (when there is no process noise is given by the second component of (16)) and is equal to $J(u_1) = x'_0 Z_1 x_0$ for controller u_1 and $J(u_2) = x'_0 Z_2 x_0$ for controller u_2 where

$$Z_1 = \begin{bmatrix} 0.5159 & -0.0314 \\ -0.0314 & 0.3257 \end{bmatrix} \quad Z_2 = \begin{bmatrix} 0.5701 & -0.0873 \\ -0.0873 & 0.3459 \end{bmatrix}.$$

When there is noise the cost of using any of the given controllers is determined by the second and third components of (16) and is equal to $J(u_1) = x'_0 Z_1 x_0 + 0.0152$ or $J(u_2) = x'_0 Z_2 x_0 + 0.0210$. Consider the point $x = [0.1, 0.1]$ for the case without any noise, u_2 is the optimal control to use, whereas with noise the optimal control to use is u_1 .

It can be clearly seen from Figures (1) and (2) that the controller switching regions depend on the process noise covariance. This differs from standard L.Q.G. optimal control where the controller gain matrix is independent of the input noise covariance for zero-mean Gaussian process noise.

3 Optimal Controller Switching with Partial Information

Let ξ be a random variable with Gaussian probability law, mean x_0 and covariance P_0 and $v(\cdot)$ is a Wiener processes independent of $w(\cdot)$ with covariance matrix $R(\cdot)$. Consider the following stochastic system whose dynamics on the interval $[0, t_N)$ are described by the stochastic differential equations

$$\begin{aligned} dx(t) &= (A(t)x(t) + B_2(t)u(t))dt + B_1(t)dw(t), \quad x(0) = \xi, \\ dy(t) &= C(t)x(t)dt + dv(t), \end{aligned} \quad (17)$$

where $x(t) \in \mathbf{R}^n$ denotes the state, $y(t) \in \mathbf{R}^p$ denotes the output and $u(t) \in \mathbf{R}^h$ is the control input. The functions $A(t)$, $B_1(t)$, $B_2(t)$, $C(t)$ are known given piecewise continuous matrix functions of time. We assume that $R(t) \geq \rho I$, $\rho > 0 \forall t \in [0, t_N)$, and we define the following two families of sigma algebras

$$\begin{aligned} \mathcal{F}^t &= \sigma(\xi, w(s), y(s), s \leq t) \\ \mathcal{Z}^t &= \sigma(y(s), s \leq t) \end{aligned} \quad (18)$$

Controller Switching with Output Feedback [12,5,4,13]. Given a collection of output feedback controllers

$$u_1(t) = K_1(t)y(t), \quad u_2(t) = K_2(t)y(t), \quad \dots, \quad u_k(t) = K_k(t)y(t) \quad (19)$$

where $K_1(\cdot), K_2(\cdot), \dots, K_k(\cdot)$ are given continuous matrix functions of time, we define a switching function $\Phi_j(\cdot)$ which is permitted to switch the controllers (19) at switching times t_j $\{j = 1, \dots, N\}$. The class of output switching functions considered in this section is of the form $\Phi_j(\cdot) : \{\mathcal{Z}^{t_j}\} \rightarrow \{1, 2, \dots, k\}$, which map from the output sigma algebra $\{\mathcal{Z}^{t_j}\}$ to the set of symbols $\{1, 2, \dots, k\}$ which index controllers (19). This class of switching functions defines the family of dynamic nonlinear output feedback controllers of the form:

$$\forall j \in \{0, 1, \dots, N-1\} \quad u(t) = K_{i_j}(t)y(t) \quad \forall t \in [t_j, t_{j+1}) \quad \text{where } i_j = \Phi_j(\mathcal{Z}^{t_j}). \quad (20)$$

The control strategy now becomes a rule for switching from one controller to another to construct a symbolic sequence $\{i_j\}_{j=0}^{N-1}$ from the sigma algebra $\mathcal{Z}^{t_j}$. Furthermore let $\mathcal{X}$ denote the set of all controls achievable by controller switching (20) with the controllers (19) adapted to $\mathcal{Z}^t$.

The solution to the partial information stochastic control problem considered in this section involves the following Ricatti differential equation

$$\begin{aligned} \dot{P}(t) &= P(t)A'(t) + A(t)P(t) - P(t)C'(t)R(t)^{-1}(t)C(t)P(t) + B_2(t)Q(t)B_2'(t) \\ \text{where } P(0) &= P_0 \end{aligned} \quad (21)$$

We consider the estimator equations of the form

$$\dot{\hat{x}}(t) = A(t)\hat{x} + P(t)C'(t)R(t)^{-1}(y(t) - C(t)\hat{x}) \quad \text{where } \hat{x}(0) = \xi \quad (22)$$

which correspond to the continuous time Kalman filter [16].

Let $S(\cdot)$ be a given function which maps from R^n to R and let $\hat{x}_0 \in R^n$ be a given vector. Introduce the following cost function:

$$W_1(t, \hat{x}, u(t)) \triangleq \mathcal{E} \left(\hat{x}'(t)N(t)\hat{x}(t) + u'(t)M(t)u(t) \right). \quad (23)$$

Then

$$F_j^i(\hat{x}_0, S(\cdot)) \triangleq \mathcal{E} (S(\hat{x}(t_{j+1}))) + \int_{t_j}^{t_{j+1}} W_1(t, \hat{x}, u_i(t), y(t))dt. \quad (24)$$

Definition 3 *The output feedback stochastic control problem with system (17) is said to have a solution via controller switching with the controllers (19) if the following conditions hold:*

- (i) *For any admissible control sequence, $u \in \mathcal{X}$, and initial condition $x(0)$ there exists a unique trajectory to the system state on the interval $[0, t_N)$ and the solution to the Riccati differential equation (21) exists and is positive definite for all $t \in [0, t_N)$.*
- (ii) *The cost given by*

$$J(u)_{u \in \mathcal{X}} = \mathcal{E} \left(\int_0^{t_N} x'(t)N(t)x(t) + u'(t)M(t)u(t)dt + x'_{t_N}X_f x_{t_N} \right) \quad (25)$$

exists and is finite for some admissible control sequence, $u \in \mathcal{X}$,

Theorem 2. *The output feedback stochastic control problem (Definition (3)) has a solution via controller switching with the output feedback controllers (19) if and only if the solution $P(\cdot)$ to the Riccati equation (21) with initial condition $P(0) = P_0$ is defined and positive definite on the interval $[0, t_N)$ and the dynamic programming equation*

$$V_N(\hat{x}_{t_N}) = \mathcal{E} (\hat{x}'_{t_N}X_f\hat{x}_{t_N}); \quad V_j(\hat{x}_{t_j}) = \min_{i=1, \dots, k} \mathcal{E} (F_j^i(\hat{x}_{t_j}, V_{j+1}(\hat{x}_{t_{j+1}}))) \quad (26)$$

has a solution for $j = 0, 1, \dots, N-1$ for all $\hat{x}_0 \in \mathbf{R}^n$.

Furthermore let $i_j(\hat{x}_0)$ be an index such that the minimum in (26) is achieved for $i = i_j(\hat{x}_0)$ and $\hat{x}(\cdot)$ be the solution to the equation (22) with initial condition $\hat{x}(0) = x_0$. Then the controller (19), (20) associated with the switching sequence $\{i_j\}_{j=0}^{N-1}$ where $i_j \triangleq \Phi_j(\hat{x}(t_j))$ solves the output feedback control problem.

Proof. (Outline) By virtue of Girsanov's theorem, Lemmata (1,2,3,4) in Section 6 below and Lemma (2.4.3) from [6] it follows that the evolution of the state of (17) with $u(t) \equiv 0$, denoted $\alpha(t)$ satisfies $\mathcal{E}^u[\alpha(t)|\mathcal{Z}^t] = \mathcal{E}^u[\alpha(t)|\mathcal{F}_u^t]$. Thus the states estimates generated by a Kalman filter are optimal when there is controller switching as considered in this paper. This is established in detail in section 6. It then follows that under the new probability measure $\mathcal{P}^u$ (defined in

section 6), the estimation error, $\epsilon \triangleq x(t) - \hat{x}(t)$ is a Gaussian process with zero mean and covariance matrix given $P(t)$ by (21). Hence, if the output feedback stochastic control problem has a solution via controller switching it follows that the solution, $P(\cdot)$, to the Ricatti differential equation (21) with initial condition, $P(0) = P_0$, is defined and positive definite on $[t_0, t_N]$. Now from Lemma (4) we have that the minimization of cost (25) is equivalent to minimizing (26) where $\hat{x}$ is the estimator state, $V(\hat{x}(t_N) = \mathcal{E}^u \hat{x}(t_N) X_f \hat{x}(t_N)$ is the cost at the final position and

$$\begin{aligned} d\hat{x}(t) &= (A(t)\hat{x} + B_2(t)u(t))dt + P(t)C'(t)R(t)^{-1}(y(t) - C(t)\hat{x})dt \\ &\triangleq (A(t)\hat{x} + B_2(t)u(t))dt + d\nu(t) \end{aligned} \quad (27)$$

where

$$\begin{aligned} \mathcal{E}^u[\nu(t)] &= 0 \\ \mathcal{E}^u[\nu'(t)\nu(t)] &= P'(t)C'(t)R(t)^{-1}C(t)P(t). \end{aligned} \quad (28)$$

The statement of the Theorem now follows immediately from Theorem 1.

Remark Results for discrete time switched controller systems can be shown to trivially follow from the results presented in this paper.

4 Illustrative Example

We consider the 2 dimensional, unstable, non-minimum phase system

$$A(t) = \begin{bmatrix} 0 & 1 \\ -1.25 & 1 \end{bmatrix}, B_1(t) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, B_2(t) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, C(t) = \begin{bmatrix} -1 \\ 2 \end{bmatrix}' \quad (29)$$

Consider the system (17) the following set of output feedback controllers

$$u_1(t) = [-1, 2]\hat{x}(t), \quad u_2(t) = [3, -6]\hat{x}(t), \quad (30)$$

and permissible controller switching at time instants jT , $j = 0, 1, \dots, 250$, $T = 0.05$ over the time interval $[0, 12.5)$ seconds. It can be easily verified that each of the controllers in (30) is independently unable to stabilize the plant.

For simulation purposes set to $X_f = 10I$ and the matrix observer Ricatti equation (21) initial condition $P(0) = I$. The process and observation noises were also set to $\mathcal{E}(w(t)) = 0$, $\mathcal{E}(w(t)w'(t)) = 0.1$, $\mathcal{E}(\nu(t)) = 0$ and $\mathcal{E}(\nu(t)\nu'(t)) = 0.1$. The matrices $N(t)$ and $M(t)$ are $N(t) = 0.1$ and

$$M(t) = \begin{bmatrix} 5 & 0 \\ 0 & 2 \end{bmatrix}. \quad (31)$$

Using the results of Theorem 2 we see in Figure (3) that by switching the two unstable controllers the system state remains within an a small neighborhood about the origin. The control input to the system, is shown in Figure (4). The switching between the two controllers is evident.

5 Conclusions

In this paper we presented a solution to certain problems in switched controller design for stochastic dynamical systems. The main result is a separation theorem for partial information systems which can be used to convert the partial information stochastic control problem to a complete information stochastic control problem. We also show that certainty equivalence does not hold and for different noise statistics the optimal switching sequence changes. Finally we show that the optimal sequence of controllers can be determined by solving a dynamic programming equation. The results presented in this paper can be extended to the infinite time interval after some technical issues are taken into consideration.

6 Appendix

There is an intrinsic difficulty in output feedback stochastic control problems arising from the fact that the controller and hence the control depends on the observation. This difficulty has been solved for linear quadratic Gaussian stochastic control problem but it is not immediately clear that these results cover the class of switching systems considered in this paper. Thus in order to establish Theorem (2), we need to establish the existence of a new probability measure in which the output and all the admissible controls are independent. This then overcomes the difficulty described above. In order to establish the existence of such a measure we will need the following definitions and Lemmata. The argument here essentially follows [6] except that special attention must be given to the switching times where solutions to the equations describing the dynamics are not well defined.

Define the processes $\alpha(\cdot)$, $\beta(\cdot)$ by

$$d\alpha = A(t)\alpha dt + dw(t), \quad \alpha(0) = \xi \quad (32)$$

$$d\beta = C(t)\beta dt + dv(t), \quad \beta(0) = 0 \quad (33)$$

also define x_1, y_1 by

$$dx_1 = (A(t)x_1 + B_2(t)u(t))dt, \quad x_1(0) = 0 \quad (34)$$

$$dy_1 = C(t)x_1 dt, \quad y_1(0) = 0. \quad (35)$$

For any admissible control $u(t)$ we define

$$\begin{aligned} x(t) &= \alpha(t) + x_1(t) \\ y(t) &= \alpha(t) + y_1(t). \end{aligned} \quad (36)$$

We then consider the following two processes [6]

$$\eta^u(t) = \exp \left[\int_0^t x'(s)C'(s)R(s)^{-1}dy - \frac{1}{2} \int_0^t x(s)'C(s)'R(s)^{-1}C(s)x(s)ds \right] \quad (37)$$

$$d^u v(t) = y(t) - \int_0^t C(s)x(s)ds \quad (38)$$

where the superscript u is to show the dependence of η^u on the the control $u(t)$.

The process η^u (38) satisfies the stochastic differential equation for all time $t \in [0, t_N) \neq t_j$ where t_j are the controller switching times

$$d\eta^u = \eta^u(x' C') R^{-1} dy, \quad \eta^u(0) = 1. \quad (39)$$

Furthermore we define $J^*(u)$ as

$$J^*(u(\cdot)) = \mathcal{E} \left[\eta^u(t_N) \left(x'(t_N) X_f x(t_N) + \int_0^{t_N} \eta^u(t) x'(t) N(t) x(t) + u(t)' M(t) u(t) \right) dt \right]. \quad (40)$$

Since the controls are piecewise linear (affine in the state) between the switching times on the interval $[t_j, t_{j+1})$ all that is required to be shown for the existence of a new probability measure $\mathcal{P}^u$ which makes the observation process independent of the control are that the cost, $J^*(u(\cdot))$ under the new measure is finite and that the Radon-Nikodym derivative [11]

$$\frac{d\mathcal{P}^u}{d\mathcal{P}} = \eta^u(t_N) \quad (41)$$

is a martingale. This is established in Lemmata (1,2,3) below which closely follow [6] but with modifications required to account for controller switching on a countable set.

Lemma 1. *There exists a constant, C_1 , independent of the control such that*

$$\mathcal{E} \eta^u(t) \mid a(t) \mid^4 \leq C_1 \quad (42)$$

Proof. Consider the following approximation of η^u ,

$$\eta_\epsilon^u(t) = \exp \left[\int_0^t x'_\epsilon C'(t) R(t)^{-1} dy - \frac{1}{2} \int_0^t (x'_\epsilon C'(t) R(t)^{-1} C(t) x_\epsilon ds) \right] \quad (43)$$

where

$$x_\epsilon(t) = \frac{x(t)}{(1 + \epsilon \mid x(t) \mid^2)^{1/2}}. \quad (44)$$

There exists a subsequence, denoted by ϵ , such that

$$\text{almost surely (a.s.) } \eta_\epsilon^u(t) \rightarrow \eta^u(t) \quad \forall t. \quad (45)$$

It follows that

$$d \mid \alpha(t) \mid^4 = 4 \mid \alpha(t) \mid^2 \alpha'(t) [A(t) \alpha(t) dt + dw] + 2 [\alpha(t)^2 tr Q(t) + 2 \alpha' Q(t) a] dt \quad (46)$$

and

$$\begin{aligned} d\eta_\epsilon^u(t)|a(t)|^4 &= 4\eta_\epsilon^u(t)|\alpha(t)|^2\alpha'(t)[A(t)\alpha(t)dt + dw] + \eta_\epsilon^u(t)|\alpha(t)|^4x'_\epsilon C'R^{-1}dy \\ &\quad + 2\eta_\epsilon^u(t)[\alpha(t)^2trQ(t) + 2\alpha'Q(t)a]dt \quad \forall t \in [0, t_N) \neq t_j. \end{aligned} \quad (47)$$

Upon integrating between $(0, t)$, using the smoothing property of integrals and taking the expectation it follows that

$$\begin{aligned} \mathcal{E}\eta_\epsilon^u(t)|\alpha(t)|^4 &= \mathcal{E}|\xi|^4 + 4\mathcal{E} \int_0^t \eta_\epsilon^u(s)|\alpha(s)|^2\alpha'(s)A(t)\alpha(t)dt \\ &\quad + 2\mathcal{E} \int_0^t \eta_\epsilon^u(s)[|\alpha(s)|^2trQ + 2a'Qa]ds. \end{aligned} \quad (48)$$

Using the fact that $\mathcal{E}\eta_\epsilon^u(s) = 1 \quad \forall s$ it can be shown that

$$\mathcal{E}\eta_\epsilon^u(t)|\alpha(t)|^4 \leq \mathcal{E}|\xi|^4 + C_2 \left(1 + \mathcal{E} \int_0^t \eta_\epsilon^u(s)|\alpha(s)|^4 ds \right) \quad (49)$$

where C_2 is a constant independent of ϵ and $u(\cdot)$. Using the Gronwall Lemma one immediately obtains

$$\mathcal{E}\eta_\epsilon^u(t)|\alpha(t)|^4 \leq C_1. \quad (50)$$

Hence taking the limit as $\epsilon \rightarrow 0$, the claim of the lemma immediately follows.

Lemma 2. *For any admissible control, $u \in \mathcal{X}$, and for all trajectories the cost denoted by $J^*(u(\cdot))$ is finite.*

Proof. It follows from the Gronwall Lemma that

$$|x_1(t)| \leq C_3 \left(\int_0^T |u(\cdot)|^2 ds \right)^{1/2}. \quad (51)$$

Using the definition of (36) and from Lemma (1) it follows that on the finite interval $[0, t_N)$ there exists a constant C_1 such that

$$\mathcal{E}\eta_\epsilon^u(t)|\alpha(t)|^2 \leq C_1. \quad (52)$$

substituting into $J^*(u(\cdot))$ the claim of the lemma is established.

Lemma 3. $\eta^u(t)$ is a $\mathcal{F}^t$ martingale.

Proof. From Lemma 2.4.2 (pg 40) of [6] all that needs to be established is that

$$\mathcal{E} \int_0^{t_N} \eta^u(t)|u(t)|^2 dt < \infty. \quad (53)$$

Using the fact that the system is piecewise linear and $\eta^u(t)$ is finite for all $t \in [0, t_N)$ it follows trivially that there exists a constant C_3 such that

$$\mathcal{E} \int_0^{t_N} \eta^u(t)|u(\cdot)|^2 dt < C_3. \quad (54)$$

The following Lemma is also needed in the proof of Theorem (2).

Lemma 4. *Suppose that the solution $P(\cdot)$ to the Ricatti differential equation (21) with initial condition $P(0) = P_0$ is defined and positive definite on the interval $[0, t_N]$ then*

$$\begin{aligned} & \min_{i=1, \dots, N} \mathcal{E}^u \left(x'(t_N) X_f x(t_N) + \int_0^{t_N} x'(t) N(t) x(t) + u'(t) M(t) u(t) dt \right) \\ &= \min_{i=1, \dots, N} \mathcal{E}^u \left(\hat{x}'(t_N) X_f \hat{x}(t_N) + \int_0^{t_N} \hat{x}'(t) N(t) \hat{x}(t) + u'(t) M(t) u(t) dt \right) \\ &+ \mathcal{E}^u \left(\epsilon'(t_N) X_f \epsilon(t_N) + \int_0^{t_N} \epsilon'(t) N(t) \epsilon(t) dt \right). \end{aligned} \quad (55)$$

Proof. Rearranging (25) we have

$$\begin{aligned} & \mathcal{E}^u \left(\mathcal{E}^u \left[x'(t_N) X_f x(t_N) + \int_0^{t_N} x'(t) N(t) x(t) + u'(t) M(t) u(t) dt \mid \mathcal{Z}^t \right] \right) = \\ & \mathcal{E}^u \left(\mathcal{E}^u \left[(x(t_N) - \hat{x}(t_N))' X_f (x(t_N) - \hat{x}(t_N)) + (x(t_N) - \hat{x}(t_N))' X_f \hat{x}(t_N) \mid \mathcal{Z}^t \right] \right) \\ & + \mathcal{E}^u \left(\mathcal{E}^u \left[\hat{x}'(t_N) X_f (x(t_N) - \hat{x}(t_N)) + \hat{x}'(t_N) X_f \hat{x}(t_N) \mid \mathcal{Z}^t \right] \right) + \\ & \mathcal{E}^u \left(\mathcal{E}^u \left[\int_0^{t_N} (x(t) - \hat{x}(t))' N(t) (x(t) - \hat{x}(t)) + \right. \right. \\ & \left. \left. (x(t) - \hat{x}(t))' N(t) \hat{x}(t) - \hat{x}'(t) N(t) (x(t) - \hat{x}(t)) \mid \mathcal{Z}^t \right] \right) + \\ & \mathcal{E}^u \left(\mathcal{E}^u \left[\int_0^{t_N} \hat{x}'(t) N(t) \hat{x}(t) + u'(t) M(t) u(t) dt \mid \mathcal{Z}^t \right] \right). \end{aligned} \quad (56)$$

Using the orthogonality property of the Kalman filter [16], namely $\mathcal{E}[(x - \hat{x})Z\hat{x}'] = 0$ where Z is any matrix, it immediately follows that

$$\begin{aligned} & \mathcal{E}^u \left(\mathcal{E}^u \left[x'(t_N) X_f x(t_N) + \int_0^{t_N} x'(t) N(t) x(t) + u'(t) M(t) u(t) dt \mid \mathcal{Z}^t \right] \right) \\ &= \mathcal{E}^u \left(\mathcal{E}^u \left[\hat{x}'(t_N) X_f \hat{x}(t_N) + \int_0^{t_N} \hat{x}'(t) N(t) \hat{x}(t) + u'(t) M(t) u(t) dt \mid \mathcal{Z}^t \right] \right) \\ &+ \mathcal{E}^u \left(\epsilon'(t_N) X_f \epsilon(t_N) + \int_0^{t_N} \epsilon'(t) N(t) \epsilon(t) dt \right). \end{aligned} \quad (57)$$

The result now follows immediately.

References

1. M.S. Branicky, V.S. Borkar, and S.K. Mitter. A unified framework for hybrid control. In *Proceedings of the 33rd IEEE Conference on Decision and Control*, pages 4228–4234, Lake Buena Vista, FL, USA, 1994.
2. R.W. Brockett. Hybrid models for motion control systems. In H.L. Trentelman and J.C. Willems, editors, *Essays in Control*. Birkhauser, Boston, 1993.
3. H. Nijmeijer and Van der Schaft. *Nonlinear dynamical control systems*. Springer-Verlag, New York, 1996.
4. A. V. Savkin, I.R. Petersen, E. Skafidas, and R.J. Evans. Hybrid dynamical systems: robust control synthesis problems. *System and Control Letters*, 29:81–90, 1996.
5. A.V. Savkin, I.R. Petersen, E. Skafidas, and R.J. Evans. Robust control via controlled switching. In *Proceedings of CESA '96*, pages 1117–1122, Lille, France, 1996.
6. A. Bensoussan. *Stochastic Control of Partially Observable Systems*. Cambridge University Press, Cambridge, 1992.
7. M.H.A. Davis. *Linear Estimation and Stochastic Control*. Chapman and Hall, London, 1978.
8. W.M. Wonham. On the separation theorem of stochastic control. *SIAM Journal of Control*, pages 312–326, 1968.
9. A.E.B. Lim, J.B. Moore, and L. Faybusovich. Separation theorem for linearly constrained "lqg" optimal control. *System and Control Letters*, 28:227–235, 1996.
10. P.R. Kumar and P. Varaiya. *Stochastic Systems: Estimation, Identification and Adaptive Control*. Prentice-Hall, Englewood Cliffs, NJ, 1986.
11. B.K. Oksendal. *Stochastic differential equations : an introduction with applications*. Springer Verlag, Berlin, 1995.
12. E. Skafidas, I.R. Petersen, R.J. Evans, and A.V. Savkin. Quadratic stabilizability of state feedback hybrid control systems. In *ICARCV '96*, pages 1073–1077, Singapore, 1996.
13. A.V. Savkin, R.J. Evans, and E. Skafidas. Absolute stabilization of uncertain continuous-time systems via discrete-event control. In *Proceedings of Workshop on Hybrid Systems*, Ithaca, New York, September, 1996.
14. D.P. Bertsekas. *Dynamic Programming*. Prentice Hall, New Jersey, 1987.
15. B. D. O. Anderson and J. B. Moore. *Optimal Control: Linear Quadratic Methods*. Prentice-Hall, 1990.
16. B.D.O. Anderson and J.B. Moore. *Optimal Filtering*. Prentice Hall, Englewood Cliffs, N.J., 1979.

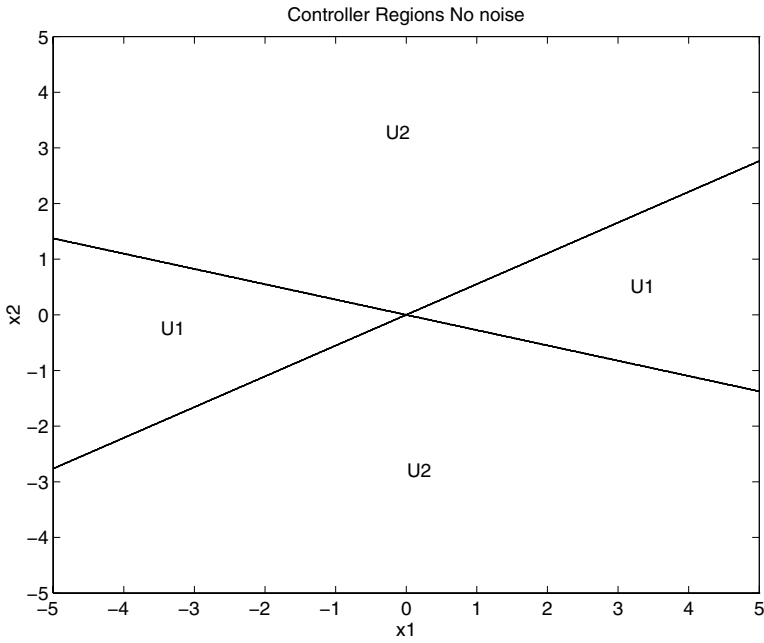


Fig. 1. Controller decision diagram with no noise.

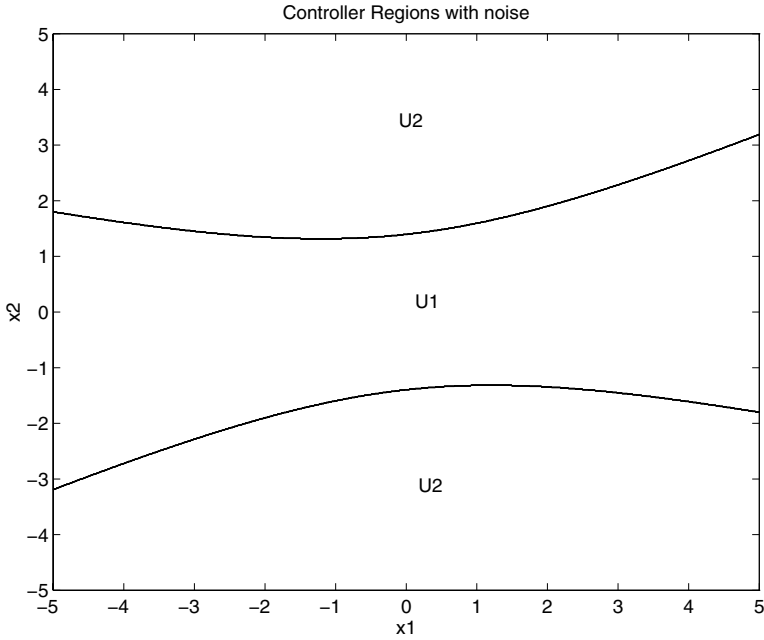


Fig. 2. Controller decision diagram with noise, $E[w] = 0, E[w'w] = 1.0$

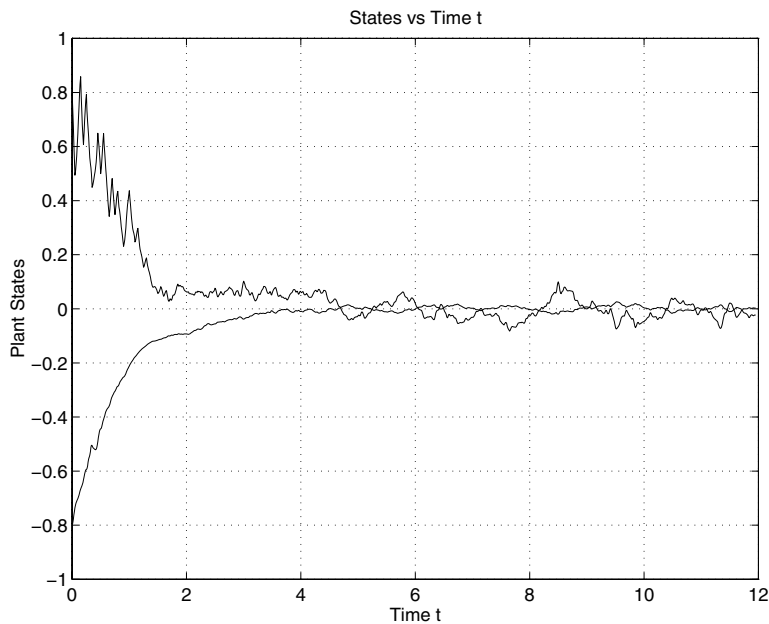


Fig. 3. Evolution of plant states with time.

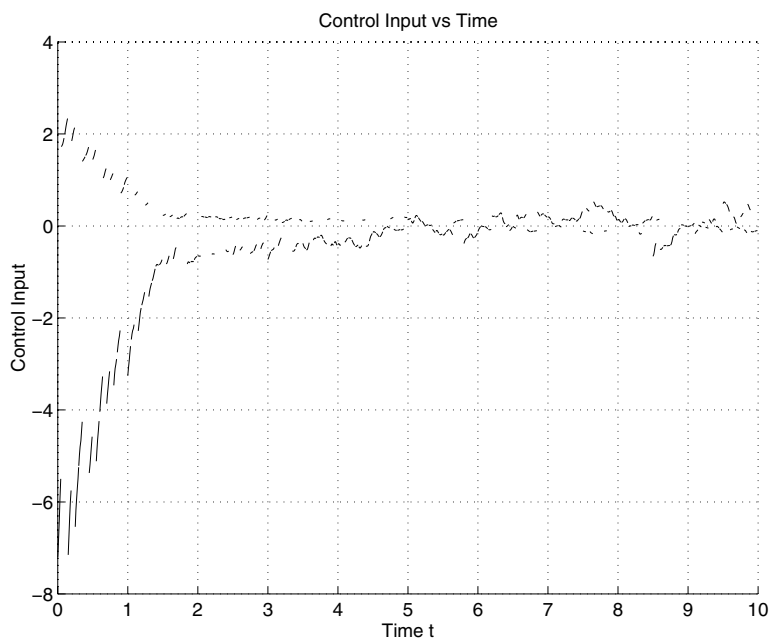


Fig. 4. Evolution of control input $u(t)$ with time for the switching controller.

Robust Control of Hybrid Systems: Performance Guided Strategies

Le Yi Wang¹, Pramod P. Khargonekar², and Ali Beydoun^{3*}

¹ Department of Electrical Engineering
Wayne State University
Detroit, Michigan 48202, USA
Tel.: 313-577-4715, Fax: 313-577-1101
lywang@ece.eng.wayne.edu

² EECS Department
University of Michigan
Ann Arbor, Michigan 48109, USA

³ Advanced Vehicle Technology
Ford Motor Company
Mail Drop 75, 21500 Oakwood Blvd.
Dearborn, Michigan 48121, USA

Abstract. In this paper, the problem of robust control of hybrid systems is investigated. The hybrid systems under study are characterized as systems whose inputs contain both analog variables and discrete actions and whose behavior belongs to an uncertainty set which includes both metric modeling errors and discrete structural uncertainties.

For a generic class of uncertain nonlinear hybrid systems, a design methodology for hybrid state feedback which maps hybrid states to hybrid control variables is developed based on system performance. Robust stability and performance of the closed-loop hybrid systems are established for systems subject to both analog modeling errors and structural uncertainties.

1 Introduction

In this paper, the problem of robust control of hybrid systems is investigated. The hybrid systems under study are characterized as systems whose inputs contain both analog variables and discrete actions and whose behavior belongs to an uncertainty set which includes both metric modeling errors and discrete structural uncertainties.

Most practical systems are hybrid in nature. These systems demonstrate the behavior of an analog dynamic system until they encounter certain abrupt structural or operating condition changes. Such changes can be triggered by unexpected causes such as component failures, or system structure variations such

* This research is supported in part by the National Science Foundation under grants ECS-9412471 and ECS-9634375, and by a research grant from Ford Motor Company.

as adding or removing of communication channels or controllers. If the transition of these operating conditions is sufficiently fast as compared to system analog dynamics, one may model these changes as instantaneous discrete switching actions of system's behavior. Associated with such discrete actions, the system changes its initial condition, dynamics, input/output variables, equilibrium points, as well as performance specifications.

It is now well recognized that there are many types of hybrid systems for which a completely unified framework may offer nothing more than an abstract language. For hybrid systems whose components are dominantly discrete events, representation theory, supervisory control, computer simulation and verification become main issues. Many of the papers in [15] [16] [17] are along this line of research. On the other hand, from the view point of classical feedback control, hybrid systems may be regarded as a case of switching control of nonlinear systems in which switching actions are taken alongside traditional analog feedback loops. Some results in this area are found in [6] [7] [10] [11]. In a generic sense, the work reported in this paper follows the main ideas of the second view point. More specifically, however, we view hybrid control as a special form of adaptation and focus on active hybrid control in which switching is a critical component of feedback design. In addition to classical analog robust methods, hybrid control is capable of adapting to large modeling errors and structural changes by tuning its control structures, on top of analog control loops.

Conceptually, hybrid control systems can potentially achieve better performance than non-switching robust controllers, due to their ability of reconfiguring and reorganizing their control structures. However, the realization of these benefits depends on a delicate coordination of discrete and analog control variables. It is well known that uncoordinated switching may cause instability for the hybrid systems, let alone performance. Some essential questions arise: How can switching decisions be made which will guarantee stability and improved performance of hybrid systems? Can hybrid systems be designed to offer robustness not only for traditional analog disturbances and metric modeling errors, but also structural uncertainties? How much switching (i.e., adaptation) is needed?

Due to complications involved in designing hybrid control systems, a brief review of existing methodologies is in order. The dwelling-time switching strategies [6] [7], in which switching is purposely slowed down to allow sufficient dwelling time for an analog controller to bring analog states close to equilibrium points, can guarantee stability. An obvious drawback is that it will lose performance by delaying switching decisions. The state space partition approaches [15] [16] [17], which first divide the whole state space into a finite set of regions and then design hybrid systems in the discrete domain, can reduce hybrid systems to finite automata. As a result, discrete-event theory can be applied. It, however, creates a daunting problem of complexity explosion and makes it difficult to address the issues of performance and robustness in the design procedures.

Recently, there are substantial research efforts in employing Lyapunov theory to analyze stability of hybrid systems [18] [19]. Since hybrid systems are special cases of nonlinear systems, this approach is certainly one of the most natural methodologies to apply. A key issue to be clarified seems to be: What are the essential and unique issues rising from hybrid systems? From this point of view, the recent work of He and Lemmon [18] is of substantial interest, which employs some intrinsic features from discrete-event part of hybrid systems to reduce the complexity of Lyapunov analysis. Since Lyapunov approaches are usually very conservative, even for linear systems, it is of essential importance to address the issues of performance in addition to stability.

The approach we propose comes from the following observations which stem from some of our work on practical hybrid systems:

1. While stability is of important concern in hybrid systems, design of active hybrid systems must demonstrate improved performance and robustness. Many practical hybrid systems are either stable or can be easily stabilized without using switching. More often than not, hybrid mechanism is introduced to improve performance and robustness, rather than nominal stability.
2. Switching decisions should be made based on potential improvement on performance. As a result, active switching decisions must be pursued which do not slow down switching decisions on purpose.
3. Switching decisions should be made based on the best utility of analog robust control methods. Hence, switching is not needed unless system performance cannot be achieved by robust analog control. This conforms to the idea of information-based adaptation, introduced by Zames [14].

In this paper, a design methodology for hybrid state feedback is introduced. The method employs traditional robust analog control for the analog state feedback, and a performance-guided switching to form a discrete state feedback. For a generic class of uncertain nonlinear hybrid systems, robust stability and performance of the closed-loop hybrid systems are established in the presence of both analog modeling errors and structural uncertainties.

The main idea of this paper stems from some simple features of classical optimal and robust control systems:

1. LQ Control: For linear time-invariant systems, optimally designed controllers via Riccati equations always guarantee stability. Hence, additional testing of Lyapunov stability is unnecessary. This fact can be extended to include robustness when LQ performance indices are expanded to include additional terms to compensate potential modelling errors.
2. H^∞ Control: Solutions to the HJI inequalities, which give sufficient conditions for H^∞ performance bounds, always guarantee input/output stability and Lyapunov stability. This is also true when modeling errors are introduced.

These basic understandings suggest that if system performance indices are appropriately selected, optimality of performance, or mere boundedness of performance, can provide stability and robustness. Potentially, this idea can be used to overcome the conservativeness of Lyapunov analysis or Lyapunov design.

The rest of the paper is organized as follows. The main structure of hybrid systems and control strategies considered in this paper is defined in Section 2. Section 3 is devoted to a more detailed formulation of problems studied in this paper. The main technical results are the performance-guided hybrid control strategies presented in Section 4. A technical condition, called performance dominant conditions, is shown to be essential for applying the strategies. Robust stability and performance of the strategies are rigorously established. The effects of using switching penalty on hybrid robustness are discussed in Section 5. Simulation results are presented alongside theoretical development to illustrate the main ideas and performance of the closed-loop hybrid systems. Finally, some conclusions are drawn in Section 6.

Some of the main ideas of this paper have been employed in practical automotive control problems [18]. While these applications indicate the usefulness of the methodology introduced in this paper, they reveal also that there are many open issues to be resolved to narrow the gap between theoretical results and practical constraints.

2 Hybrid Control Systems

2.1 Hybrid Systems

Hybrid systems studied in this paper, as depicted as in Figure 1, are input/output systems whose inputs contain both analog variables and discrete actions. Analog variables take real values in their respective intervals, and discrete actions can assume only a finite number of values. Consequently, one may label discrete actions with a finite set of symbols.

In the absence of discrete actions, i.e., when the system is in a fixed discrete state s , a hybrid system resembles an analog input/output system. Hence, its evolution behavior can be reasonably modelled by traditional mathematical models for dynamic systems, such as differential or difference equations, transfer functions, etc.

When a discrete action is activated, the evolution of the system is interrupted, and its behavior undergoes an abrupt change. The change may entail a switching in system structure, a jump in internal states, a reconfiguration of inputs and outputs, a shift of equilibrium points, as well as installation of new design specifications. This abrupt change is summarized by saying that the system is switched from the old discrete state s to a new one s_1 . Consequently, we can represent discrete actions by $\sigma = (s, s_1)$.

Traditionally, analog inputs are decomposed into (uncontrolled) disturbances d and control variables u . Similarly, discrete actions can be divided into (uncontrolled) discrete uncertainty δ and controlled discrete actions α . Further, outputs will be grouped into measured outputs y_m and unmeasured ones y_u . When necessary, we will use σ to denote both δ and α ; and use y to denote both y_m and y_u .

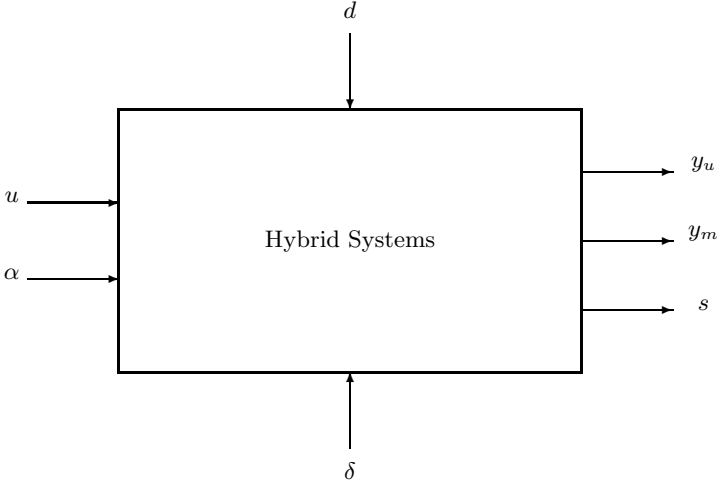


Fig. 1. Hybrid Systems

Mathematically, a hybrid system is a mapping

$$(s, y) = \mathbf{H}(u, \alpha; d, \delta).$$

The mapping is not completely known. Generically, the information about $\mathbf{H}$ is given by an uncertainty set Ω which contains $\mathbf{H}$. Ω will be specified in the subsequent sections.

2.2 Models

At a given discrete state s , the system output is related to the analog inputs u and d via an uncertain mapping

$$y = \mathbf{G}_s(u, d)$$

where the information about $\mathbf{G}_s$ is given by the uncertainty set Ω_s which contains $\mathbf{G}_s$.

A discrete (either controlled or uncontrolled) action $\sigma = (s, s_1)$ at t causes the switching of the system from state s to state s_1

$$y = \mathbf{G}_{s_1}(u, d)$$

with a new initial value $y(t_+)$ which may be different from $y(t_-)$. Observe that in addition to the observed jumps from $y(t_-)$ to $y(t_+)$, the switching may also result in an unmeasured internal disturbance. Here, the effect of switching disturbance on the system is lumped as part of d . Of course, in the special case where the entire analog state x is measured, i.e., $y = x$, such jumps will be all included in changes from $y(t_-)$ to $y(t_+)$.

Evolution of the discrete state s can be modelled as a discrete-event system in which discrete modes are represented by *discrete states* and discrete actions as *discrete events*. Transition and evolution of discrete states can then be modelled by an automaton.

2.3 Hybrid Control

Hybrid control studied in this paper is limited to causal mappings from measured output y_m and discrete state s to controlled analog variable u and discrete action α

$$(u, \alpha) = \Phi(y_m, s)$$

such that the resultant closed-loop system

$$(s, y) = \mathbf{H}(\Phi(y_m, s); d, \delta)$$

demonstrates desired behavior in the presence of d , δ and uncertainty Ω .

Generally speaking, design of such hybrid systems includes the following components.

Analog Feedback: Robust Stability and Performance Analog feedback is responsible for determining analog control strategies $u = \mathbf{F}_s(y_m)$ at a given discrete state s for the system $y = \mathbf{G}_s(u, d)$ so that robustness, stability and performance of the analog closed-loop subsystem $y = \mathbf{G}_s(\mathbf{F}_s(y_m), d)$ can be achieved in the presence of model uncertainty and disturbances. These are conventional systems described by differential equations, difference equations, frequency responses, etc., and controlled by using robust, optimal or adaptive control methodologies. It will be shown that the essential information which must be communicated to discrete feedback is the achievable robust performance for the subsystem.

DES Supervisors: Enabled Event Sets At a given discrete state s , DES supervisors determine which events are to be disabled. As a result, the output of the supervisors is a set $\Sigma_{DES}(s)$ containing all the events deemed admissible by the supervisors. This set must be available to the discrete feedback for making switching decisions.

Discrete Feedback: Constrained Prioritization Discrete feedback is responsible for making switching decisions. More specifically, at any given time the discrete feedback must determine controlled discrete actions based on information submitted from hybrid states, DES and analog systems. Symbolically, the discrete state feedback is simply a constrained prioritization:

$$\inf_{\sigma \in \Sigma_c(s)} \mathcal{P}(\sigma)$$

where σ is the discrete event which is to be selected from the constrained set $\Sigma_c(s)$ at the current discrete state s . The function $\mathcal{P}(\sigma)$ is a priority function which assigns priority orders to different event commands σ . Note that $\mathcal{P}$ is not necessarily numerical. It only requires a relation $\leq$. For instance, $\sigma_1 \leq \sigma_2$ will simply mean that in the evaluation process, the event σ_1 is judged as proceeding σ_2 , and hence is a better choice. As a result, the main task for establishing the discrete feedback is to construct the constrained event set $\Sigma_c(s)$ and priority function $\mathcal{P}$.

1. Constrained Event Sets

Usually, for a given discrete state s , the constrained set $\Sigma_c(s)$ can be constructed from certain conditions. Suppose at the state s , the unconstrained set of all possible events is given by

$$\Sigma(s) = \{\varepsilon, \sigma_1 = (s, s_1), \dots, \sigma_n = (s, s_n)\}$$

where ε is the null event (i.e., no discrete switching occurs). Obviously, $\Sigma_c(s) \subseteq \Sigma(s)$.

Examples of possible constraints which reduce $\Sigma(s)$ to $\Sigma_c(s)$ are numerous. We will list some typical cases one often encounters in practical problems.

- Discrete Constraints. This is provided by DES supervisors. At the current discrete state s , the DES supervisors provide a set $\Sigma_{DES}(s)$ of enabled events. Consequently, $\Sigma_c(s) \subseteq \Sigma_{DES}(s)$.
- Analog Constraints. For example, at time t , analog constraints may be expressed as the conditions: “If the measured output $y_m(\tau)$ belongs to a set Γ , then $\sigma_1 = (s, s_1)$ is disabled.”
- Hybrid Constraints. Such constraints will depend on both discrete and analog states. Take, for example, the hypothetical condition in an automotive control problem: “if $gear(t-T) = 1$ and $gear(t) = 2$ and vehicle speed $v(t) > 20$, then the shift $(2, 1)$ is not allowed,” where T is a small time interval. In practice, this constraint will prevent gear chattering between Gear 1 and Gear 2. Apparently, this constraint mandates the memory of the discrete state s . Another possible case is given by the constraint “if $gear(t) = 1$ and $v \geq 40$ or $\dot{v} \geq c > 0$, then the nonswitching decision $\varepsilon = (1, 1)$ is not allowed.” This reflects

the fact that staying at Gear 1 is not desired unless the vehicle is cruising at low speeds.

The issue of constructing $\Sigma_c(s)$ is currently under investigation and will not be covered in this paper.

2. Priority Functions

The constrained set $\Sigma_c(s)$ provides the freedom for the discrete feedback to make switching decisions at state s . The primary responsibility of the discrete feedback is then to make sound decisions within the set. This responsibility can be formulated by using the priority function $\mathcal{P}$, which rank orders the events in $\Sigma_c(s)$ on the basis of stability, system performance, robustness, coordination with other systems, environment conditions, etc. When the task can be defined and evaluated off-line, the priority function can be constructed *a priori*, and implementation of the discrete feedback becomes a “simple” finite optimization procedure. In the more complicated, and more realistic, situations where priority functions rely on environment conditions, priority functions must be constructed on-line, and the design of the discrete feedback can become much more involved.

In this paper, we will focus on the design of discrete feedback. We will demonstrate that when system performance measures satisfy certain dominant conditions, they become viable candidates as priority functions in designing discrete feedback.

3 Problem Formulation

While practical systems seldom allow full state measurements and state feedback, it is beneficial for understanding achievable performance and design limitations to study first the “simpler” cases of state feedback. This can be viewed as a special case of our general formulation by identifying the measured output y_m with the state x , $y_m = x$. Technical results of this paper are limited to design of hybrid state feedback.

In this and subsequent sections we will need the following basic notation.

$\mathbf{R}, \mathbf{C}, \mathbf{Z}$ denote the real numbers, complex numbers and integers, respectively. The absolute value of $x \in \mathbf{C}$ is $|x|$, and the real part of x is $\Re(x)$.

For a vector $x \in \mathbf{R}^n$, its Euclidean norm will be denoted by $\|x\|$. For a matrix $M \in \mathbf{R}^{n \times m}$, its largest and smallest singular values are denoted by $\bar{\sigma}(M)$ and $\underline{\sigma}(M)$, respectively.

l^∞ denotes the spaces of sequences $\{u(t) \in \mathbf{R}^n, t \in \mathbf{Z}\}$, for which

$$\|u\|_\infty = \sup_{t \in \mathbf{Z}} \|u(t)\| < \infty.$$

For a time $t \in \mathbf{R}$, t_- and t_+ will denote its left and right limits, respectively. Hence, $x(t_-)$ is the left limit of x at t .

3.1 Hybrid Plants In State Space with Uncertainties

In this paper, we are considering hybrid systems expressed in the state-space form

$$\begin{aligned}\dot{x} &= f(x, u, d; s; t) + \Delta(x, u, d; s; t), \\ s &= DES(\delta, \alpha)\end{aligned}\tag{1}$$

where $x(t) \in \mathbb{R}^n$ is the analog state, $u(t) \in \mathbb{R}^m$ the analog control input, $d \in \Omega_d$ the analog disturbances; $s \in \mathbf{Q}$ is the discrete state, $\delta \in \Omega_\delta$ is the uncontrolled discrete action. Ω_δ is called *discrete uncertainty set*. $\alpha(t) \in \Sigma$ is the discrete control action, and Σ is a finite set of admissible discrete control actions. $\Delta \in \Omega$ represents model uncertainties. Ω is called *model uncertainty set*. f is assumed to be continuous in x and u . DES is a discrete event automaton.

The discrete action σ (i.e., δ or α) takes the form of $\sigma = (s, s_1)$ where s is the old discrete state and s_1 the new one. Associated with a discrete action at t , the system nominal dynamics will switch from $\dot{x} = f(x, u, d; s; t)$ to $\dot{x} = f(x, u, d; s_1; t)$, and the state x will jump from $x(t_-)$ to $x(t_+)$.

It is well known that switching among stable systems may result in instability if switching is not carefully coordinated, as demonstrated in the following simple example.

Example 1 Consider two linear systems

$$\dot{x} = A_s x + B_s u, \quad s = 1, 2.$$

Suppose that for a given discrete state s , a state feedback K_s is designed such that the corresponding closed-loop system

$$\dot{x} = (A_s - B_s K_s)x$$

is asymptotically stable. It appears that for this system, the discrete event system should have only two states: $s = 1$ and $s = 2$, and both states are stable ones. However, it is easy to construct examples in which there exist (infinitely) many switching sequences which will result in instability, even though both subsystems are stable individually.

For example, suppose

$$A_1 - B_1 K_1 = \begin{bmatrix} -100 & 20 \\ 200 & -100 \end{bmatrix}; \quad A_2 - B_2 K_2 = \begin{bmatrix} -100 & 200 \\ 20 & -100 \end{bmatrix}.$$

The two closed-loop subsystems are stable individually. However, when switching between the two systems occurs at kT , $k = 1, 2, \dots$, with $T = 0.01$, and $x(0) = [1 \ 0]^T$, the resultant hybrid system is unstable, as shown in Figure 2.

This example suggests that analog control and discrete actions must be coordinated to achieve ultimate goals of robust stability and performance.

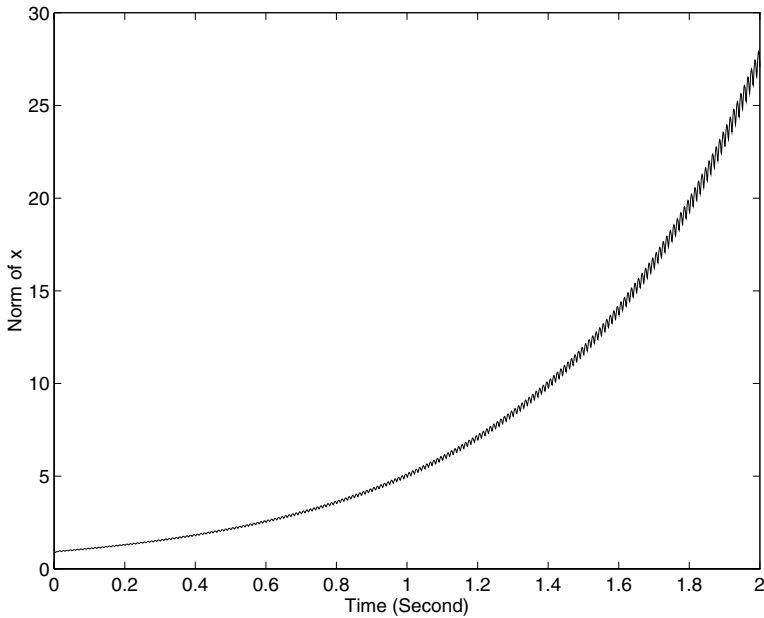


Fig. 2. An Unstable Hybrid System

3.2 Robust Hybrid Control Problems

Suppose a hybrid system is modeled by (1). Without loss of generality, we assume

$$f(0, 0, 0; s; t) = 0, \quad \forall s \in \mathbb{Q},$$

namely, $x = 0, u = 0$ is an equilibrium point of the nominal system.

At the current time t , the performance index for expected future performance is expressed generically as

$$J_t(x, u) = \int_t^\infty w(\tau - t)h(x(\tau), u(\tau), \tau)d\tau \quad (2)$$

where $h(x, u, t) \geq 0$ is a penalty function on analog state and control variables. The moving-window weighting function $w(t)$ represents a tradeoff between transient and persistent performance. More often than not, active switching is used in practical systems to achieve better transient performance (e.g., gear shifting to meet an acceleration command and better fuel economy). As a result, $w(t), t > 0$, is often a monotone decreasing function of time, although this is not an essential assumption for theoretical development.

Penalty on switching will be introduced later in the design process. Switching penalty does not enter the system performance index directly. But rather, it will become either a constraint or an additional term in switching decisions to achieve

tradeoff among system performance, switching frequency, robustness, etc. While it has been well established in classical control problems that performance indices shall penalize deviations of x and u from 0 (the equilibrium point under study), a reasonable penalty on the discrete action α is often its switching rather than its discrete states before or after the switching. For instance, to avoid busy shifting of transmission gears, the shifting, rather than gear positions, should be penalized.

Definition 1 A *hybrid state feedback* consists of two mappings which relate the discrete actions and analog control to analog and discrete states:

$$\alpha = (s, s_1) = \phi(x, s), \quad u(t) = g(x, s),$$

where $g(0, s) = 0$ for all $s \in \mathbf{Q}$. ϕ and g will be termed as *discrete state feedback* and *analog state feedback*, respectively.

After applying a designed hybrid state feedback, the closed-loop hybrid system can be expressed as

$$\begin{aligned} \dot{x} &= f(x, g(x, s), d; s; t) + \Delta(x, g(x, s), d; s; t), \\ s &= DES(\delta, \phi(x, s)) \end{aligned} \tag{3}$$

The assumption $g(0, s) = 0$ implies that the equilibrium point of the nominal closed-loop system under study is $x = 0$.

Problem For the system (1) and a given performance level μ , design a discrete state feedback ϕ and an analog state feedback g such that for all $\Delta \in \Omega$, $d \in \Omega_d$, and $\delta \in \Omega_\delta$ the equilibrium point $x = 0$ of the closed-loop system (3) is stable (in the sense to be defined later) and the system achieves the uniform performance level μ

$$\sup_t J_t(x, g(x, s), \phi(x, s)) \leq \mu.$$

4 Analysis of Hybrid Systems

Usually, practical hybrid systems employ active switching to improve system performance. This is exemplified by automotive transmissions, VDE engines, swirl control, hybrid vehicles, etc. Consequently, one may be tempted to use directly the actual practical performance measures, say fuel economy, to determine switching actions. A potential danger of such strategies is that switching based on these performance measures may actually lead to a loss of stability. A central question is: What type of performance measures implies stability? Performance

dominant conditions introduced below define a class of performance measures which imply stability.

It should be emphasized here that the notion of stability we employ in this paper is not Lyapunov stability. Consequently, the results presented in this paper should not be confused with those from multiple Lyapunov approaches. One exception is the case of LQ hybrid design for which Lyapunov stability can also be established.

4.1 Performance Dominance Conditions

In the subsequent subsections we will introduce some performance guided hybrid control strategies. The strategies rely on the fact that if performance measures have certain dominance features, uniform boundedness of them will guarantee the uniform boundedness of analog states. As a result, coordination of hybrid control variables can achieve transient performance and boundedness of signals (stability) simultaneously, without explicitly limiting switching rates to guarantee stability. This will allow active switching actions, as often required by practical hybrid control problems. Hence, we will first introduce a key lemma which explains conditions under which performance measures become dominant over stability.

Consider an analog nonlinear time-varying system in state space form

$$\dot{x}(t) = f(x, t) \quad (4)$$

where $x(t) \in \mathbb{R}^n$ is the state variable. To accommodate possible switching actions in hybrid systems, $f(x, t)$ is assumed to be only piece-wise continuous in t . Suppose a performance measure is imposed on the system

$$J(t) = \int_t^\infty h(x(\tau), \tau) d\tau$$

where $h(x, t) \geq 0$.

Definition 2 Performance Dominant Conditions

The performance measure J is said to be *dominant* for the system (4) if there exist constants $c, k_1, k_2 \geq 0$ such that for all $t \geq 0$

$$\|f(x(t), t)\| \leq k_1 h(x(t), t) + c, \text{ when } \|f(x(t), t)\| \geq 1, \quad (5)$$

$$\|x(t)\| \leq k_2 h(x(t), t), \text{ when } \|x(t)\| \geq 1. \quad (6)$$

Essentially, these conditions stipulate that:

1. $h(x, t)$ dominates the rate $\dot{x}$ of state evolution. As a result, any finite escaping of states will certainly show up in the performance measure J .

2. Large persistent x values must be detected in J , avoiding the situation in which the tail of x remains large even when J is bounded.

The main utility of performance dominance is elaborated in Lemma 1.

Lemma 1 *Suppose J is dominant for the system (4). If*

$$\sup_{t \geq 0} J(t) \leq \mu < \infty,$$

then

$$\|x\|_{\infty} \leq (k_1 + k_2)\mu + 2 + c. \quad (7)$$

Proof: Suppose at some time $t \geq 0$, $\|x(t)\| = M$. We will show that $M \leq (k_1 + k_2)\mu + 2 + c$.

For any $\tau \in [t, t+1]$,

$$x(t) = x(\tau) - \int_t^{\tau} f(x(\theta), \theta) d\theta.$$

It follows that

$$\begin{aligned} \|x(t)\| &\leq \|x(\tau)\| + \int_t^{\tau} \|f(x(\theta), \theta)\| d\theta \\ &\leq \|x(\tau)\| + \int_t^1 \|f(x(\theta), \theta)\| d\theta. \end{aligned}$$

Let $\Omega_1 = \{\theta \in [t, t+1] : \|f(x(\theta), \theta)\| < 1\}$ and $\Omega_2 = [t, t+1] \ominus \Omega_1$. Then, by (5)

$$\begin{aligned} \|x(t)\| &\leq \|x(\tau)\| + \int_{\Omega_1} d\theta + \int_{\Omega_2} [k_1 h(x(\theta), \theta) + c] d\theta \\ &\leq \|x(\tau)\| + 1 + c + k_1 J(t) \\ &\leq \|x(\tau)\| + 1 + c + k_1 \mu \end{aligned}$$

by (6). Consequently, by taking $\int_t^{t+1} d\tau$ on both sides, noting that $\|x(\tau)\| \leq 1 + k_2 h(x(\tau), \tau)$, $\forall x$, we obtain

$$\begin{aligned} \|x(t)\| &\leq \int_t^{t+1} (\|x(\tau)\| + 1 + c + k_1 \mu) d\tau \\ &\leq \int_t^{t+1} [k_2 h(x(\tau), \tau) + 1] d\tau + 1 + c + k_1 \mu \\ &\leq k_2 \mu + 1 + 1 + c + k_1 \mu. \end{aligned}$$

Since the right hand side is independent of t , we have

$$\|x\|_{\infty} \leq (k_1 + k_2)\mu + 2 + c$$

as required. □

The notion of stability given by (7), namely boundedness of state variables, is not a Lyapunov stability. This is more closely related to the notion of stability used in adaptive control. This is not a surprise, considering that switching control is a special form of adaptation. It should also be pointed out that additional conditions can be introduced if asymptotic stability of x is required. This is shown in Lemma 2.

Example 2 It is noted that for linear time-varying systems $\dot{x} = A(t)x$ with $\sup_{t \geq 0} \bar{\sigma}(A(t)) \leq k$, all quadratic measures $J(t) = \int_t^\infty x^T Q x dt$, with $Q > 0$, are dominant with $c = 0$.

Indeed, for $\|x\| \geq 1$, $x^T Q x \geq \underline{\sigma}^2(Q) \|x\|^2 \geq \underline{\sigma}^2(Q) \|x\|$. That is,

$$\|x\| \leq \frac{1}{\underline{\sigma}^2(Q)} x^T Q x.$$

Furthermore, the condition $\|A(t)x(t)\| \geq 1$ implies $1 \leq \|A(t)x(t)\| \leq k\|x\|$, or $\|x\| \geq \frac{1}{k}$. Hence,

$$\|x\| \leq \frac{k}{\underline{\sigma}^2(Q)} x^T Q x.$$

Consequently,

$$\|A(t)x(t)\| \leq k\|x\| \leq \frac{k}{\underline{\sigma}^2(Q)} x^T Q x.$$

It follows that

$$x^T Q x \geq \underline{\sigma}^2(Q) \|x\|^2 \geq \frac{\underline{\sigma}^2(Q)}{k} \|x\|.$$

4.2 Performance Dominance Conditions for Hybrid Systems

Consider a hybrid system consisting of l nonlinear nominal subsystems

$$\dot{x}(t) = f(x(t), u(t); s; t), \quad s = 1, \dots, l, \quad (8)$$

where f is continuous in x and u .

Suppose for each subsystem, an analog bounded nonlinear state feedback

$$u = g(x, s), \quad s = 1, \dots, l$$

with $\|g(x, s)\| \leq k_s \|x\|$, has been designed such that $x = 0$ of the s -th closed-loop subsystem

$$\dot{x} = f(x, g(x, s); s; t) \quad (9)$$

is asymptotically stable. Furthermore, at time t a performance measure

$$J(t) = \int_t^\infty w(\tau - t) h(x, u, \tau) d\tau$$

is imposed for design of the hybrid system, where $h(x, u, \tau) \geq 0$ and the moving-window weighting function $w(t)$ satisfies $0 < w_l \leq w(t) \leq w_h, \forall t \geq 0$.

First, we will show that the weighting function w , which is often selected to reflect tradeoff between transient and persistent performance, will not affect the dominance of a performance index.

Lemma 2 *Suppose that in the special case of unweighted performance, i.e., $w = 1$, J is dominant for (9) for all $s = 1, \dots, l$. If a switching strategy results in a uniformly bounded performance*

$$\sup_{t \geq 0} J(t) \leq \eta < \infty,$$

then the state x of the closed-loop hybrid system is bounded by

$$\|x\|_\infty \leq \frac{(k_1 + k_2)}{w_l} \mu + 2 + c. \quad (10)$$

If in addition there exists an integer $m > 0$ and a constant $k > 0$ for which

$$\|x(t)\|^m \leq kh(x, u, t) \quad (11)$$

then $x(t) \rightarrow 0$, as $t \rightarrow \infty$.

It should be pointed out that Lemma 2 is slightly different from asymptotic Lyapunov stability which requires continuous dependence of $\|x\|_\infty$ on $\|x(0)\|$. The notion of stability here resembles those used in adaptive systems where asymptotic stability is interpreted as boundedness and convergence of state variables. Unless specified otherwise, this notion of asymptotic stability will be employed in this paper.

It is noted in passing that for linear quadratic problems in Example 2, the conditions of Lemma 2 are satisfied with $c = 0$ and $m = 2$, since for all x , $x^T Q x \geq \underline{\sigma}^2(Q) \|x\|^2$ when $\underline{\sigma}(Q) > 0$.

Proof: We first observe that after applying the analog feedback, we arrive at the closed-loop analog subsystem

$$\dot{x} = f(x, g(x, s); s; t).$$

Since the unweighted J is dominant, for some $c, k_1, k_2 \geq 0$

$$\|f(x, u; s; t)\| \leq k_1 h(x, u, t) + c \leq \frac{k_1}{w_l} w(\tau - t) h(x, u, t) + c, \text{ if } \|f(x, u; s; t)\| \geq 1$$

$$\|x(t)\| \leq k_2 h(x, u, t) \leq \frac{k_1}{w_l} w(\tau - t) h(x, u, t), \text{ if } \|x(t)\| \geq 1.$$

As a result, the weighted performance $J(t)$ is still dominant for the closed-loop systems. Now by Lemma 1, the uniform boundedness of $J(t)$ implies that

$$\|x\|_\infty \leq \frac{(k_1 + k_2)}{w_l} \mu + 2 + c.$$

The conclusion $x(t) \rightarrow 0$, as $t \rightarrow \infty$ follows from the fact that

$$\begin{aligned} \int_0^\infty \|x(t)\|^m dt &\leq \int_0^\infty kh(x, g(x), t) dt \\ &\leq \frac{k}{w_l} \int_0^\infty w(t)h(x, g(x), t) dt < \infty. \end{aligned}$$

□

Lemma 2 reveals that under the performance dominant conditions, if a hybrid control strategy results in a uniformly bounded performance J , then stability or asymptotic stability of the closed-loop hybrid system is automatically guaranteed, independent of the choice of weighting functions, provided that they are uniformly bounded away from 0. This will be important in hybrid control problems, where weighting functions must be tuned to reflect desired tradeoff between transient and persistent performance. In other words, switching strategies which improve the performance measure $J(t)$ will always imply stability. This will allow a design of hybrid state feedback without explicitly limiting switching rates.

5 Performance Guided Hybrid Feedback

Lemma 2 suggests a natural hybrid control strategy which employs $J(t)$ as a guideline in the switching decision of discrete actions.

5.1 General Results

A hybrid control must specify both analog and discrete feedback mappings. In this paper, we will focus on the design of discrete state feedback, i.e., switching decisions. Hence, we will assume that for each subsystem, an analog robust state feedback has been designed.

Performance Guided Hybrid Feedback:

1. The Analog Feedback: Suppose that for each discrete state $s \in \mathbf{Q}$, an analog bounded state feedback

$$u(t) = g(x(t), s)$$

is applied such that $x = 0$ of the s -th closed-loop subsystem

$$\dot{x} = f(x, g(x, s); s; t)$$

is asymptotically stable.

2. The Discrete Feedback: Suppose that at t and analog state $x(t_-)$, the constrained discrete action set for s is $\Sigma_c(s)$. Other than the singular case $\Sigma_c(s) = \{\varepsilon\}$ (i.e., no switching is allowed), the discrete feedback must decide whether a discrete action should be activated. Let

$$J(x(t), s; t) = \int_t^\infty w(\tau - t)h(x, g(x, s), \tau)d\tau,$$

i.e., the expected future performance when the discrete state remains at s after t . Suppose that a discrete action $\alpha = (s, s_1) \in \Sigma_c(s)$ will result in a state jump from $x(t_-)$ to $x(t_+)$. Then, at time t , the discrete action $\alpha = (s, s_1)$ is determined by

$$(s, s_1) = \phi(x(t_-), s) = \arg \min \{J(x(t_+), s_1; t) : (s, s_1) \in \Sigma_c(s)\}. \quad (12)$$

In other words, the priority function for designing discrete feedback is expressed as: $\alpha_1 = (s, s_1) \in \Sigma_c(s)$ proceeds $\alpha_2 = (s, s_2) \in \Sigma_c(s)$, i.e., $\alpha_1 < \alpha_2$, if

$$J(x(t_+), s_1, t) < J(x(t_+), s_2, t).$$

Hence, α_1 is a better choice than α_2 if switching to s_1 provides a better expected performance than switching to s_2 . In the competing case

$$J(x(t_+), s_1, t) = J(x(t_+), s_2, t),$$

one may select any one for switching.¹

Note that we do not explicitly limit the magnitude of the state jumps. The key consideration here is their effects on achievable performance rather than their metric distance.

Define

$$\eta_0 = J(x(t_-, s_0), s_0, t).$$

η_0 is the performance achieved by using the non-switching control strategy at the discrete state s_0 for all time after t_- .

Theorem 1 *If the performance index J is dominant for each closed-loop subsystem*

$$\dot{x} = f(x, g(x, s); s; t)$$

and satisfies (11), then the performance guided hybrid feedback guarantees

¹ The competing case may be resolved by using other criteria which are transparent in our formulation.

1. $\|x\|_\infty$ is bounded, and $x(t) \rightarrow 0$, as $t \rightarrow \infty$.
2. At any time t , the resulting performance $J(t)$ of the hybrid control strategy is better than or equal to non-switching control, namely,

$$J(t) \leq \eta_0. \quad (13)$$

Proof: Suppose that the switching sequence is $\tilde{s}$. Then, the system equation along this trajectory is given by

$$\dot{x} = f(x, g(x, \tilde{s}); \tilde{s}; t) =: f_0(x, t).$$

By definition, the performance-guided hybrid feedback implies that at any time t ,

$$J(t) = \min_s J(x, s, t) \leq J(x, \tilde{s}, t) = \eta_0$$

which implies (13).

Now, the performance dominant conditions for all subsystem imply that

$$\begin{aligned} \|f_0(x, t)\| &\leq \max_s \|f(x, s, t)\| \\ &\leq kh(x, t) + c, \text{ when } \|f_0(x, t)\| \geq 1 \end{aligned}$$

for some $k > 0$ and $c \geq 0$. Consequently, all conditions of Lemma 2 are satisfied and the first conclusion follows. $\square$

5.2 Hybrid Linear Quadratic Control

In the special case where the underlying analog systems are linear time invariant and performance measures are LQ types, explicit computations can be easily obtained. In fact the stronger results of asymptotic Lyapunov stability hold in this case. For this reason and practical importance of LQ problems, we provide detailed discussions on hybrid LQ design in this subsection.

Consider a hybrid system which consists of l linear subsystems

$$\dot{x} = A_s x + B_s u, \quad s = 1, 2, \dots, l,$$

where $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, and all pairs (A_s, B_s) are controllable.

A quadratic performance index is given by

$$J(t) = \int_t^\infty [x^T(\tau)Qx(\tau) + u^T(\tau)Ru(\tau)]d\tau$$

where Q and R are real, symmetric and positive definite. If there is no switching, it is well known that for the s -th subsystem the optimal control, which minimizes

J , can be obtained from the positive definite solutions K_s to the algebraic Riccati equation:

$$K_s A_s + A_s^T K_s + Q - K_s B_s R^{-1} B_s^T K_s = 0$$

and the corresponding optimal linear state feedback is

$$u(\tau) = -R^{-1} B_s^T K_s x(t), \quad \tau \geq t.$$

The optimal performace can also be easily computed as

$$J_s(t) = \frac{1}{2} x^T(t) K_s x(t).$$

Obviously, if no switching occurs, the s -th closed-loop subsystem

$$\dot{x} = (A_s - B_s R^{-1} B_s^T K_s) x = \tilde{A}_s x$$

is asymptotically stable. Hence, all eigenvalues $\lambda_j(\tilde{A}_s)$ of $\tilde{A}_s$ have negative real part. Let

$$\sigma_s = \max_{j=1, \dots, n} |\Re(\lambda_j(\tilde{A}_s))|.$$

It follows that there exists a constant $c_s > 0$ such that for any initial state $x(0)$

$$\|x(t)\| \geq c_s e^{-\sigma_s t} \|x(0)\|. \quad (14)$$

Denote

$$\begin{aligned} \sigma &= \max_{s=1, \dots, l} \sigma_s > 0 \\ c &= \min_{i=1, \dots, l} c_s > 0. \end{aligned}$$

Then,

$$\|x(t)\| \geq c_s e^{-\sigma_s t} \|x(0)\|.$$

Intuitively, this inequality means that $x(t)$ cannot decrease to 0 extremely fast. As a result, if $\inf_t^{t+T} x^T(\tau) Q x(\tau) d\tau$ is small, $\max_{\tau \in [t, t+T]} \|x(\tau)\|$ cannot be too big. Hence, boundedness of J will imply boundedness of $\|x\|$.²

Suppose a sequence of discrete actions occur at

$$t_0 < t_1 < t_2 < \dots$$

Lemma 3 *For any given switching sequence, if the corresponding performance index is bounded*

$$J(t) = \int_t^\infty [x^T(\tau) Q x(\tau) + u^T(\tau) R u(\tau)] d\tau \leq \eta < \infty, \quad \forall t$$

then the equilibrium point $x = 0$ of the hybrid system is globally asymptotically Lyapunov stable.

² One can easily construct a case of nonlinear systems in which extremely large values of $x(t)$ can escape the detection of J when $x(t)$ can decrease to 0 extremely fast.

Proof: First, $J(t) \leq \eta, \forall t$ implies

$$\int_0^\infty x^T(t)Qx(t)dt \leq \eta.$$

Since $Q > 0$, for $a = \underline{\alpha}^2(Q)$

$$\eta \geq \int_0^\infty x^T(t)Qx(t)dt \geq a \int_0^\infty x^T(t)x(t)dt.$$

Hence,

$$\int_0^\infty x^T(t)x(t)dt \leq \frac{\eta}{a} := \eta_1. \quad (15)$$

We will show by contradiction that

$$\|x\|_\infty = \sup_t \|x(t)\| \leq \frac{1}{c} \sqrt{\frac{2\eta_1\sigma}{1-e^{-2\sigma}}}.$$

Hence, suppose that for some $t \geq 0$, $\|x(t)\| = M > \frac{1}{c} \sqrt{\frac{2\eta_1\sigma}{1-e^{-2\sigma}}}$. Then for $\tau \in [t, t+1]$, by (14) we have

$$\begin{aligned} \|x(\tau)\|^2 &\geq c^2 e^{-2\sigma(\tau-t)} \|x(t)\|^2 \\ &\geq c^2 e^{-2\sigma(\tau-t)} M^2. \end{aligned}$$

It follows that

$$\begin{aligned} \int_t^{t+1} x^T(\tau)x(\tau)d\tau &\geq c^2 M^2 \int_t^{t+1} e^{-2\sigma(\tau-t)} d\tau \\ &= \frac{c^2 M^2}{2\sigma} [1 - e^{-2\sigma}]. \end{aligned}$$

However, by (15), $\int_t^{t+1} x^T(\tau)x(\tau)d\tau \leq \eta_1$, i.e.,

$$\frac{c^2 M^2}{2\sigma} [1 - e^{-2\sigma}] \leq \eta_1$$

or

$$M \leq \frac{1}{c} \sqrt{\frac{2\eta_1\sigma}{1-e^{-2\sigma}}}.$$

This contradicts the hypothesis. Therefore, we conclude that

$$\|x\|_\infty \leq \frac{1}{c} \sqrt{\frac{2\eta_1\sigma}{1-e^{-2\sigma}}}. \quad (16)$$

Since for LQ problems η_1 is always bounded by

$$\eta_1 = \frac{\eta}{a} \leq \frac{\kappa \|x(0)\|^2}{a}$$

for some constant $\kappa > 0$, the bound (16) implies that

$$\|x\|_\infty \leq \frac{1}{c} \sqrt{\frac{2\kappa\sigma}{a(1 - e^{-2\sigma})}} \|x(0)\|$$

and $x = 0$ is Lyapunov stable.

Finally, it is obvious that $J(t) \leq \eta < \infty, \forall t$ implies $x(t) \rightarrow 0$, as $t \rightarrow \infty$, since $Q > 0$. □

Lemma 3 shows that any discrete switching sequences which produce uniformly bounded performance J will automatically guarantee asymptotic Lyapunov stability. Since this result does not require a minimum residing or dwelling time on any subsystems, fast switching can be accommodated. In particular, the following performance guided hybrid control will guarantee a stable closed-loop hybrid system.

Performance Guided Hybrid Feedback:

1. The Analog Feedback: If the current discrete state is s , then

$$u(t) = -R^{-1}B_s^T K_s x(t).$$

2. The Discrete Feedback: Suppose that the constrained discrete action set at s is given by $\Sigma_c(s)$. At time t , if the current analog state is $x(t)$, then the discrete action $\alpha = (s, s_1)$ is determined by

$$s_1 = \arg \min \{x^T(t) K_{s_1} x(t) : (s, s_1) \in \Sigma_c(s)\}. \quad (17)$$

Define

$$\eta_0 = \min \{x^T(0) K_s x(0) : s = 1, \dots, l\}.$$

η_0 is the best performance achievable by using LQ non-switching control strategies.

Theorem 2 *The performance guided hybrid feedback guarantees that*

1. $x = 0$ of the closed-loop hybrid system is globally asymptotically stable in the Lyapunov sense.
2. The resultant performance $J(0)$ of the hybrid control strategy is never worse than the best non-switching control,

$$J(0) \leq \eta_0. \quad (18)$$

Proof:

By Lemma 3, we only need to show that any switching sequence produced by the feedback mappings will guarantee that

$$J = \int_0^\infty [x^T(t)Qx(t) + u^T(t)Ru(t)]dt \leq \eta_0 < \infty.$$

Suppose that the discrete state sequence is

$$s = \{s(t_j), \quad j = 0, 1, \dots\}, \quad t_0 = 0.$$

For any integer $N > 0$, the truncated switching sequence is defined as

$$s_N = \{s(0), s(t_1), \dots, s(t_N)\}.$$

The corresponding performance index is given by

$$\begin{aligned} J_N &= \int_0^{t_1} + \dots + \int_{t_{N \otimes 1}}^{t_N} + \int_{t_N}^\infty \\ &= \int_0^{t_1} + \dots + \int_{t_{N \otimes 1}}^{t_N} + x^T(t_N)K_{s(t_N)}x(t_N). \end{aligned}$$

By the switching criterion (17),

$$\begin{aligned} J_N &\leq \int_0^{t_1} + \dots + \int_{t_{N \otimes 1}}^{t_N} + x^T(t_N)K_{s(t_{N \otimes 1})}x(t_N) \\ &= \int_0^{t_1} + \dots + \int_{t_{N \otimes 2}}^{t_{N \otimes 1}} + x^T(t_{N-1})K_{s(t_{N \otimes 1})}x(t_{N-1}). \end{aligned}$$

Further iterations lead to

$$J_N \leq x^T(0)K_{s(0)}x(0) = \eta_0, \quad (19)$$

noting that $s(0)$ is selected according to (17).

Since the right hand side of (19) is independent of N , we conclude that

$$J(0) = \lim_{N \rightarrow \infty} J_N \leq \eta_0$$

as stated. □

Remarks: In concrete examples, it can be expected that the actual $J(0)$ can be much better than η_0 , that is, switching is useful for improving performance. However, there are cases of hybrid systems in which non-switching control is in fact optimal. For instance, if there exists s_0 such that $K_{s_0} - K_s \leq 0$, $s = 1, \dots, l$, then the non-switching control with $s(t) \equiv s_0$, $t \geq 0$ is the optimal choice.

6 Switching Penalty and Hybrid Robustness

A key property which was employed to establish Theorems 1 and 2 is the monotone decreasing (non-increasing) of the performance index, alluded in the design of discrete feedback. The property is only valid if the plant model is exactly known and all discrete actions are control variables. For a plant modelled by (1) with both discrete uncertainty and analog modeling errors, the discrete feedback cannot guarantee the monotonicity of J . Consequently, uniform boundedness of $J(t)$ may not hold, and hence stability cannot be guaranteed. We are therefore seeking modified switching strategies, which will guarantee *robust* stability and performance, in the presence of analog modeling errors and discrete uncertainties.

6.1 Systems with Analog Uncertainty

Consider a hybrid system with analog modeling errors

$$\dot{x} = f(x, u; s; t) + \Delta(x, u; s; t), \quad s = 1, \dots, l \quad (20)$$

where $\Delta \in \Omega_s$ is a term of analog uncertainty at the discrete state s . While the actual form or value of Δ is unknown, the nominal system f and the uncertainty sets Ω_s , $s = 1, \dots, l$, are given *a priori*. At time t with the analog state $x(t)$, the performance index for the system is given by

$$J(x(t), t) = \int_t^\infty w(\tau - t)h(x, u, \tau)d\tau.$$

Suppose that for each discrete state s , an analog feedback has been designed, $u(t) = g(x(t), s)$, which achieves robust stability for the s -th subsystem (20).

For a given analog state $x(t)$ at t and a discrete state sequence $\{s(\tau_i) : t \leq \tau_1 < \tau_2 < \dots\}$, the nominal trajectory $x^*(\tau)$, $\tau \geq t$ is the one governed by the nominal hybrid system

$$\dot{x}^*(\tau) = f(x^*(\tau), g(x^*(\tau), s(\tau)); s(\tau); \tau)$$

with the initial condition $x^*(t) = x(t)$. The resultant nominal performance is given by

$$J^*(x(t), t) = \int_t^\infty w(\tau - t)h(x^*(\tau), g(x^*(\tau), s(\tau)), \tau)d\tau.$$

In the special case when no switching occurs after t , we denote for $s = 1, \dots, l$,

$$J_s^*(x(t), t) = \int_t^\infty w(\tau - t)h(x^*(\tau), g(x^*(\tau), s), \tau)d\tau.$$

Modeling errors Δ will perturb the system trajectories from the nominal ones. When the analog feedback mappings g are designed robustly, the closed-loop subsystem

$$\dot{x} = f(x, g(x, s); s; t) + \Delta(x, g(x, s); s; t)$$

is asymptotically stable, and the corresponding performance value for the non-switching strategy is bounded

$$J_s(x(t), t) = \int_t^\infty w(\tau - t)h(x(\tau), g(x(\tau), s), \tau)d\tau < \infty$$

for all $\Delta \in \Omega_s$. It is observed that $J_s(x(t), t)$ cannot be determined *a priori* since it depends on Δ . As a result, it cannot be used as an index for design.

On the other hand, the nominal performance index J_s^* and the worst-case performance errors

$$e_s(x(t), t) = \sup_{\Delta \in \Omega_s} |J_s(x(t), t) - J_s^*(x(t), t)|, \quad s = 1, \dots, l$$

are available at t , and can be employed in the design process.³

Hence, we introduce the following construction of discrete feedback mappings.

Performance Guided Hybrid Feedback with Switching Penalty:

1. The Analog Feedback: If the current discrete state is s , then

$$u(t) = g(x(t), s).$$

2. The Discrete Feedback: Suppose at t_- the discrete state is s with the constrained discrete action set $\Sigma_c(s)$. The priority function is defined as follows. At time t_+ , the new discrete state s_1 is determined by

$$s_1 = \arg \min \{J_{s_1}^*(x(t); t) + \text{SP}(s, s_1) : (s, s_1) \in \Sigma_c(s)\} \quad (21)$$

where the switching penalty term $\text{SP}(s, s_1)$ is defined by

$$\text{SP}(s, s_1) = \begin{cases} 0 & , \quad s_1 = s \\ e_s(x(t_+)) + e_{s_1}(x(t_+)) + \varepsilon_0 & , \quad s_1 \neq s \end{cases}$$

and $\varepsilon_0 > 0$ is a constant.

³ It should be emphasized here that “availability” of $e_s(x(t), t)$ is of information nature. Namely, the information is available for computing $e_s(x, t)$. Actual computational complexity of $e_s(x, t)$ will obviously depend on the nonlinear system, uncertainty sets, performance measures, etc. These computational issues are not addressed in this paper.

Intuitively, introduction of the switching penalty term will impose cautiousness in switching discrete states. More precisely, a switching from the current discrete state s to the new s_1 is selected only if the nominal performance level $J_{s_1}^*(x(t_+), t_+)$ is better than $J_s^*(x(t_+), t_+) - e_{s_1}(x(t_+), t_+) - e_s(x(t_+), t_+)$, namely

$$J_{s_1}^*(x(t_+), t_+) + e_{s_1}(x(t_+), t_+) < J_s^*(x(t_+), t_+) - e_s(x(t_+), t_+).$$

In other words, switching will be decided only if the worst-case performance for the system s_1 is better than the best performance of the system s . The following theorem establishes the robustness of this strategy.

Theorem 3 *Suppose that the performance index J is dominant for each subsystem. The performance guided hybrid feedback with switching penalty guarantees*

1. *Robust asymptotic stability of the closed-loop hybrid system.*
2. *The resulting performance J of the hybrid control strategy is bounded by the robust worst-case performance of non-switching control strategies, namely,*

$$J \leq \min_{s \in \mathbf{Q}} [J_s^*(x(0), 0) + e_s(x(0), 0)].$$

Proof: The proof is similar to that for Lemma 2 and is omitted. □

Example 3 Consider a hybrid system with six discrete states,

$$\dot{x} = A_s x + B_s u, \quad s = 1, 2, 3, 4, 5, 6$$

where

$$\begin{aligned} A_1 &= \begin{bmatrix} 1 & -5 \\ 9 & -1 \end{bmatrix}, & B_1 &= \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \\ A_2 &= \begin{bmatrix} 1 & 2 \\ 2 & 6 \end{bmatrix}, & B_2 &= \begin{bmatrix} 2 \\ 4 \end{bmatrix}, \\ A_3 &= \begin{bmatrix} 0.7 & 6 \\ 0.9 & -5.88 \end{bmatrix}, & B_3 &= \begin{bmatrix} 1 \\ 1.5 \end{bmatrix}, \\ A_4 &= \begin{bmatrix} 1 & -7.8 \\ 2.05 & -5.56 \end{bmatrix}, & B_4 &= \begin{bmatrix} 0.5 \\ 0.4 \end{bmatrix}, \\ A_5 &= \begin{bmatrix} 3 & 9.1 \\ 1.9 & 1.2 \end{bmatrix}, & B_5 &= \begin{bmatrix} 1 \\ -5 \end{bmatrix}, \\ A_6 &= \begin{bmatrix} 2 & 2.5 \\ 2 & 2.7 \end{bmatrix}, & B_6 &= \begin{bmatrix} 2 \\ 1.1 \end{bmatrix}. \end{aligned}$$

The analog state feedback gains K_s are designed by solving the standard algebraic Riccati equations with performance weighting matrices $Q = \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix}$ and

$R = 0.1$. Switching penalties are defined by the matrix SP:

$$\text{SP} = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix} \times d,$$

where d is a scaling constant. Here, $\text{SP}(s, s_1)$ is the penalty for switching from discrete state s to discrete state s_1 .

For $d = 0.06$ (heavy switching penalty) and $d = 0.001$ (light switching penalty), simulation results are presented in Figures 3 and 4. The optimal non-switching performance is compared to the performance guided switching strategy. Apparently, in this case the switching strategy offers a much better performance. It is observed that increasing switching penalty will result in infrequent switching (good) and a loss of performance (bad), which seems to be an understandable tradeoff.

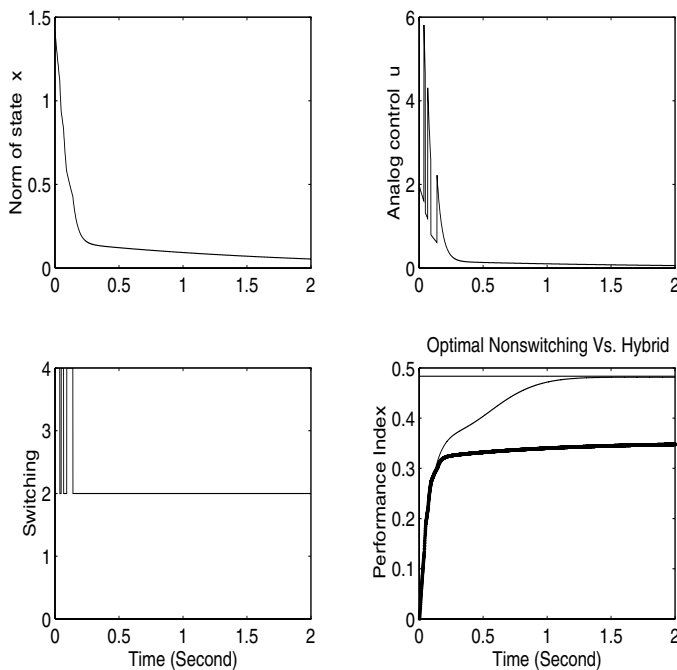


Fig. 3. Hybrid Control With Switching Penalty: $d = 0.06$

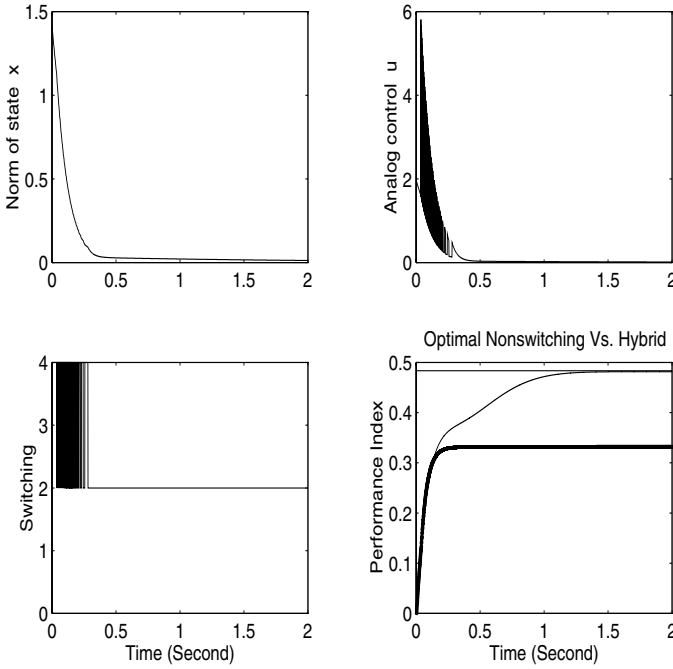


Fig. 4. Hybrid Control With Switching Penalty: $d = 0.001$

6.2 Systems with Discrete Uncertainties

Analysis of Robust Stability For stability analysis, one may lump both δ and α as a discrete action σ . Consider an uncertain nonlinear hybrid system

$$\dot{x} = f(x, u; s; t) + \Delta(x, u; s; t), \quad s = 1, 2, \dots, l, \quad (22)$$

where $\Delta \in \Omega_s$ is the modeling uncertainty.

Suppose that for each discrete state s , a (nonlinear) state feedback $u = g(x, s)$ has been designed such that the closed-loop system

$$\dot{x} = f(x, g(x, s); s; t) + \Delta(x, g(x, s); s; t), \quad s = 1, 2, \dots, l \quad (23)$$

is robustly Lyapunov stable. More precisely, we assume that there exist Lyapunov functions $V_s(x)$ for the s -th subsystem which satisfies that for some $a_s > 0$, $b_s > 0$, $c_s > 0$

$$a_s \|x\|^2 \leq V_s(x) \leq b_s \|x\|^2$$

and

$$\frac{dV_s(x)}{dt} \leq -c_s \|x\|^2$$

along the trajectories of the system (22) for all $\Delta \in \Omega_s$. There are now numerous robust design methods for nonlinear systems which will guarantee robust Lyapunov stability. The reader is referred to [5][9] for more reference.

Denote

$$\gamma_s = \frac{b_s}{a_s}, \quad \text{and} \quad \gamma_0 = \max_{s=1, \dots, l} \gamma_s.$$

First, we establish a property of discrete sequences under which asymptotic stability of the hybrid system is guaranteed. The reader is referred to [18] [19] for related stability results.

Lemma 4 *Suppose a discrete state sequence $\{s(t), \quad t \geq 0\}$ has its switching actions occur at $t_i, i = 0, 1, \dots$ with $t_0 < t_1 < t_2 < \dots$. If for some $0 < \beta < 1$,*

$$\|x((t_i)_+)\|^2 \leq \beta \|x((t_{i-1})_+)\|^2, \quad (24)$$

then the equilibrium point $x = 0$ of the hybrid system is robustly asymptotically stable in the Lyapunov sense.

Proof: The hypothesis implies that after switching

$$\|x((t_i)_+)\|^2 \leq \beta^i \|x((t_0)_+)\|^2, \quad i = 1, 2, \dots$$

Hence,

$$x((t_i)_+) \rightarrow 0, \quad i \rightarrow \infty.$$

Furthermore, for $t \in (t_i, t_{i+1})$ the discrete state is $s = s((t_i)_+)$ and

$$\begin{aligned} \|x(t)\|^2 &\leq \frac{1}{a_s} V_s(x(t)) \\ &\leq \frac{1}{a_s} V_s(x((t_i)_+)) \quad \text{since } V_s \text{ is a Lyapunov function of the subsystem } s \\ &\leq \frac{b_s}{a_s} \|x((t_i)_+)\|^2 \\ &\leq \gamma_0 \|x((t_i)_+)\|^2 \\ &\leq \gamma_0 \beta^i \|x((t_0)_+)\|^2. \end{aligned}$$

Since $0 < \beta < 1$,

$$\|x(t)\|^2 \leq \gamma_0 \|x((t_0)_+)\|^2, \quad \forall t \in [t_0, \infty)$$

and

$$x(t) \rightarrow 0, \quad \text{as } t \rightarrow \infty.$$

These imply that $x = 0$ of the hybrid system is asymptotically stable in the Lyapunov sense. □

As a result of Lemma 4, to analyze asymptotic stability of a hybrid control scheme one only needs to establish conditions on discrete sequences $s(t_i)$, $i = 0, 1, \dots$, such that the inequality (24) is valid.

For a selected $0 < \eta < 1$, define

$$T_s = \frac{(\gamma_0 - \eta) a_s}{\eta c_s}$$

and

$$T = \max\{T_s : s = 1, \dots, l\}.$$

Theorem 4 *Suppose that the switching will cause bounded state jumps, uniformly for all switching actions. Namely, for some constant κ*

$$\|x(t_+)\|^2 \leq \kappa \|x(t_-)\|^2,$$

at switching time t . If $\beta = \kappa\eta < 1$ and the discrete state sequence $s(t_i)$, $i = 0, 1, \dots$ satisfies

$$t_i - t_{i-1} \geq T, \quad i = 1, 2, \dots,$$

then the equilibrium point $x = 0$ of the hybrid system is robustly asymptotically stable in the Lyapunov sense.

Proof: By Lemma 4 we only need to establish the condition (24) for the given sequence. In the interval (t_i, t_{i+1}) , the discrete state is $s = s((t_i)_+)$. By the hypothesis on the s -th subsystem, we have

$$V_s(x((t_{i+1})_-)) \leq V_s(x((t_i)_+)) - c_s \int_{t_i}^{t_{i+1}} x^T(t)x(t)dt. \quad (25)$$

First, we show that if at any $t \in (t_i, t_{i+1})$

$$\|x(t)\|^2 \leq \frac{\eta}{\gamma_0} \|x((t_i)_+)\|^2,$$

then the condition (24) is satisfied.

Indeed, in this case for any $t \in (t_i, t_{i+1})$ we have

$$\begin{aligned} \|x((t_{i+1})_-)\|^2 &\leq \frac{1}{a_s} V_s(x((t_{i+1})_-)) \\ &\leq \frac{1}{a_s} V_s(x(t)) \\ &\leq \frac{b_s}{a_s} \|x(t)\|^2 \\ &\leq \gamma_0 \frac{\eta}{\gamma_0} \|x((t_i)_+)\|^2 \\ &\leq \eta \|x((t_i)_+)\|^2. \end{aligned}$$

It follows that

$$\|x((t_{i+1})_+)\|^2 \leq \kappa \|x((t_{i+1})_-)\|^2 \leq \kappa \eta \|x((t_i)_+)\|^2 \leq \beta \|x((t_i)_+)\|^2.$$

Hence, we only need to consider the case where for all $t \in (t_i, t_{i+1})$

$$\|x(t)\|^2 > \frac{\eta}{\gamma_0} \|x(t_i)\|^2.$$

It follows that

$$\begin{aligned} \|x((t_{i+1})_-)\|^2 &\leq \frac{1}{a_s} V_s(x((t_{i+1})_-)) \\ &\leq \frac{1}{a_s} [V_s(x((t_i)_+)) - c_s \int_{t_i}^{t_{i+1}} x^T(t)x(t)dt] \quad \text{by (25)} \\ &\leq \gamma_0 \|x((t_i)_+)\|^2 - \frac{c_s}{a_s} \int_{t_i}^{t_{i+1}} x^T(t)x(t)dt \\ &\leq \gamma_0 \|x((t_i)_+)\|^2 - \frac{c_s}{a_s} \frac{\eta}{\gamma_0} \|x((t_i)_+)\|^2 (t_{i+1} - t_i) \\ &= [\gamma_0 - \frac{c_s}{a_s} \eta (t_{i+1} - t_i)] \|x((t_i)_+)\|^2. \end{aligned}$$

Therefore,

$$\|x((t_{i+1})_+)\|^2 \leq \kappa [\gamma_0 - \frac{c_s}{a_s} \eta (t_{i+1} - t_i)] \|x((t_i)_+)\|^2.$$

Clearly, the condition (24) will be satisfied if

$$\gamma_0 - \frac{c_s}{a_s} \eta (t_{i+1} - t_i) \leq \eta,$$

or equivalently

$$(t_{i+1} - t_i) \geq \frac{\gamma_0 - \eta}{\frac{c_s}{a_s} \eta}.$$

Therefore, the hypothesis $(t_{i+1} - t_i) \geq T$ implies the condition (24). By Lemma 4, $x = 0$ of the hybrid system is asymptotically stable in the Lyapunov sense. □

Theorem 4 claims that if each subsystem is well designed to achieve robust asymptotic stability individually, then the total hybrid system will remain asymptotically stable if switching causes only bounded state jumps and does not occur very frequently. This is the same idea as the dwelling time concept for logic switching in adaptation introduced by Morse [6].

Example 4 Using the same system as in Example 1, i.e.,

$$\dot{x} = A_s x, \quad s = 1, 2$$

where

$$A_1 = \begin{bmatrix} -100 & 20 \\ 200 & -100 \end{bmatrix}; \quad A_2 = \begin{bmatrix} -100 & 200 \\ 20 & -100 \end{bmatrix}.$$

We can solve the Lyapunov equations

$$A_1 P_1 + P_1 A_1^T = -I; \quad A_2 P_2 + P_2 A_2^T = -I$$

to obtain

$$P_1 = \begin{bmatrix} 0.0068 & 0.0092 \\ 0.0092 & 0.0233 \end{bmatrix}; \quad P_2 = \begin{bmatrix} 0.023 & 0.0092 \\ 0.0092 & 0.0068 \end{bmatrix}.$$

Then

$$b = \overline{\sigma}(P_1) = \overline{\sigma}(P_2) = 0.0274; \quad a = \underline{\sigma}(P_1) = \underline{\sigma}(P_2) = 0.0028.$$

Also, $Q = I$ and $c = \underline{c}(Q) = 1$. It follows that $\gamma_0 = 9.7857$.

Now, for a selected, say, $\beta = 0.9$, we have

$$\begin{aligned} T_0 &= \frac{\gamma_0 - \beta}{\beta} \frac{a}{c} \\ &= \frac{9.7857 - 0.9}{0.9} \frac{0.0028}{1} \\ &= 0.0278. \end{aligned}$$

In this case, any switching sequences which cause state jumps bounded by

$$\|x(t_+)\|^2 < \frac{1}{0.9} \|x(t_-)\|^2$$

and occur at least T_0 apart will guarantee asymptotic stability.

Figure 5 demonstrates the hybrid system for switching at kT , $k = 1, 2, \dots$, $T = 0.04$, starting at $x(0) = [1 \ 0]^T$, without state jumps at switching.

Theorem 4 is useful in applications where infrequent switching is either physically valid (component failures), acceptable (air conditioning switching, cruise control switching in automotive systems), or desirable (slow adaptation for slowly varying plants). It is particularly useful in “passive” or “lazy” hybrid systems in which discrete actions are not control variables. In such hybrid systems, one can employ the theorem for analyzing stability or constraining switching intervals.

Discrete Robustness When a hybrid system contains uncontrolled discrete actions, control strategies must often be modified to accommodate uncertainties in discrete actions. Uncontrolled discrete actions can be further decomposed into two classes: Measured and unmeasured actions. Unmeasured discrete actions are those whose occurrence cannot be directly sensed or deduced uniquely from other measured quantities. Since these actions cannot be detected uniquely, control strategies must be designed so that these discrete actions will not result in

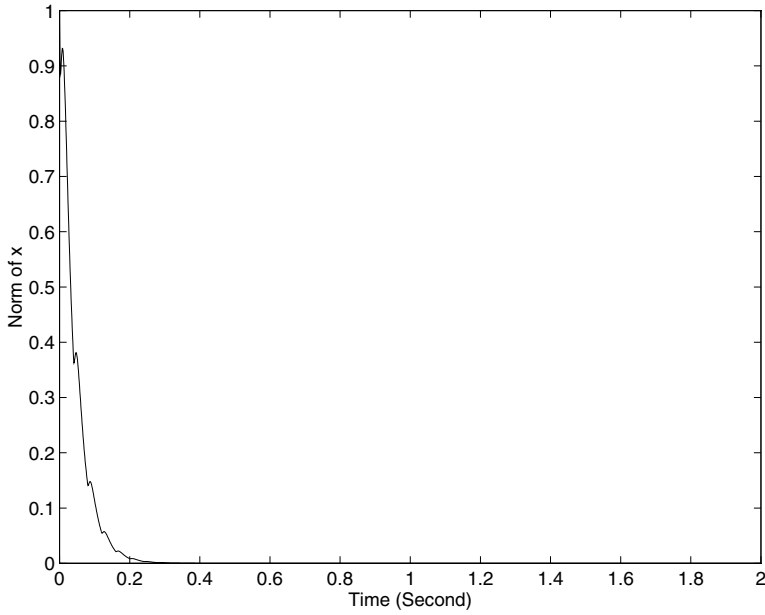


Fig. 5. A Stable Hybrid System

instability or total system failures. On the other hand, the measurable discrete actions can either be measured directly or deduced uniquely from measured variables. For this type of uncontrolled actions, switching schemes can be devised so that adverse effects from these actions can be corrected, at least partially, by adjusting either analog mapping u or discrete action α . Typical examples of unmeasured discrete actions include component failures in circuit boards, battery cell malfunctioning in electrical vehicle power supplies, etc. Measured actions are exemplified by cruise control switching which can happen any time but is known to the powertrain control system.

Mathematically, we will consider a hybrid system modelled in state space form

$$\begin{aligned}\dot{x} &= f(x, u; s; t) \\ s &= DES(\delta_u, \delta_m, \alpha)\end{aligned}$$

where $\delta_u \in \Omega_u$ is an unmeasured and uncontrolled discrete action; $\delta_m \in \Omega_m$ is a measured and uncontrolled discrete action; and $\alpha \in \Sigma$ is the discrete control action.

Suppose for each given measured discrete uncertainty $\delta_m \in \Omega_m$, a discrete feedback

$$\alpha = \phi(x, s)$$

and an analog state feedback

$$u = g(x, s)$$

have been designed such that the resulting closed-loop subsystem

$$\begin{aligned}\dot{x} &= f(x, g(x, s); s; t) \\ s &= DES(\delta_u, \delta_m, \phi(x, s))\end{aligned}\tag{26}$$

is Lyapunov stable, in the sense of Section 5. The system (26) can be expressed concisely as

$$\begin{aligned}\dot{x} &= \tilde{f}(x; s; t) \\ s &= DES(\delta, x)\end{aligned}$$

where $\delta \in \Omega_d$ is a generically labelled discrete action representing all possible combinations of δ_u and δ_m .

By Lemma 4, to analyse asymptotic stability of the system (26) we may establish conditions on discrete state sequences $s(t_i)$, $i = 0, 1, \dots$, such that the inequality (24) is valid.

Define for some $0 < \eta < 1$,

$$T_s = \frac{(\gamma_s - \eta)}{\eta} \frac{b_s}{a_s}$$

and

$$T = \max\{T_s : s \in \mathbf{Q}\}.$$

Theorem 5 *If the uncontrolled switching δ only causes bounded state jumps*

$$\|x(t_+)\|^2 \leq \kappa \|x(t_-)\|^2$$

with $\beta = \kappa\eta < 1$ and the discrete uncertainty sequence $\delta(t_i)$, $i = 0, 1, \dots$ satisfies

$$t_i - t_{i-1} \geq T, \quad i = 1, 2, \dots,$$

then the equilibrium point $x = 0$ of the hybrid system (26) is asymptotically stable in the Lyapunov sense.

Proof: The proof is the same as that of Theorem 4. □

Since δ is uncontrolled, it cannot be actively selected. Therefore, Theorem 5 is essentially a “passive” conclusion as Theorem 4. It claims that when analog and discrete feedback mappings are appropriately designed to achieve robust stability, the hybrid system will remain stable in the presence of discrete uncertainties, provided the uncertain discrete actions neither occur too frequently nor cause unbounded state jumps.

7 Conclusions

The basic hybrid control structure and performance-guided hybrid control strategies introduced in this paper provide a framework in which stability, performance and robustness of hybrid systems can be analyzed and constructive design of hybrid feedback can be performed. This paper does not give any detailed treatment of constructing the constrained set of enabled discrete events, which is an important part of the total design procedure. This is currently under investigation.

References

1. M.S. Branicky, V. Borkar and S.K. Mitter, A unified framework for hybrid control, *Proc. of IEEE CDC Conference*, pp. 4228-4234, 1994.
2. R.W. Brockett, Hybrid models for motion control systems, *Essays on Control: Perspectives in the Theory and its Applications*, Tentleman and Willems, eds, pp. 29-54, 1993.
3. P.E. Caines and Y. Wei, Hierarchical hybrid control systems, *Notes from Workshop on Control Using Logic-Based Switching*, Block Island, RI, pp. 50-57, 1995.
4. M. Chang and E.J. Davison, Switching control of a family of plants, *Proc. of 1995 ACC*, pp. 1015-1020, 1995.
5. H.K. Khalil, *Nonlinear Systems*, 2nd. Ed., Prentice Hall, 1996.
6. A.S. Morse, Supervisory control of families of linear set-point controllers, *Proc. the 32nd CDC Conference*, pp. 1055-1060, 1993.
7. A.S. Morse, Dwell time switching, *Proc. the 2nd ECC Conference*, pp. 176-181, 1993.
8. F.M. Pait and A.S. Morse, A cyclic switching strategy for parameter adaptive control, *IEEE Trans. Automat. Control*, Vol. AC-39, pp. 1172-1183, 1994.
9. H. Nijmeijer and A.J. van der Schaft, *Nonlinear Dynamic Control Systems*, Springer, 1990.
10. K. Poolla and J.S. Shamma, Optimal asymptotic robust performance through logic based switching, *Notes from Workshop on Control Using Logic-Based Switching*, Block Island, RI, pp. 44-49, 1995.
11. P.J. Ramadge, Some issues in rule-based switching control, *Workshop on Control Using Logic-Based Switching*, Block Island, RI, 1995.
12. L.Y. Wang, A. Beydoun, J. Cook, J. Sun and I. Kolmanovsky, Optimal hybrid control with automotive applications, in *Logic-Based Switching Control*, A.S. Morse (ED.), Springer, 1996.
13. J.C. Willems, *The analysis of feedback systems*, MIT Press, Cambridge, 1971.
14. G.Zames, "Information, complexity and adaptation," *Control Using Logic-Based Switching*, ed. A.S. Morse, Springer-Verlag, 1996.
15. A.S. Morse (Ed.), *Control Using Logic-Based Switching*, Springer Verlag, LNCIS 222, 1996.
16. G.L. Grossman, A. Nerode, A.P. Ravn and H. Rischel (Eds.), *Hybrid Systems*, Springer Verlag, LNCS 736, 1993.
17. P. Antsaklis, W. Kohn, A. Nerode and S. Sastry (Eds.), *Hybrid Systems II*, Springer Verlag, LNCS 999, 1995.

18. T. A. Henzinger and S. Sastry (Eds.), Hybrid Systems: Computation and Control, Springer Verlag, LNCS 1386, 1998.
19. Special issue on hybrid control systems, *IEEE Trans. Automat. Control*, Vol. AC-43, April 1998.