Logics for Hybrid Systems

J. M. DAVOREN, MEMBER, IEEE, AND ANIL NERODE, MEMBER, IEEE

Invited Paper

Hybrid systems are heterogenous dynamical systems characterized by interacting continuous and discrete dynamics. Such mathematical models have proved fruitful in a great diversity of engineering applications, including air-traffic control, automated manufacturing, and chemical process control. The high-profile and safety-critical nature of the application areas has fostered a large and growing body of work on formal methods for hybrid systems: mathematical logics, computational models and methods, and computer-aided reasoning tools supporting the formal specification and verification of performance requirements for hybrid systems, and the design and synthesis of control programs for hybrid systems that are provably correct with respect to formal specifications. This paper offers a synthetic overview of, and original contributions to, the use of logics and formal methods in the analysis of hybrid systems.

Keywords*—Automata, computer-aided analysis, computer-aided software engineering, design automation, formal languages, hybrid control systems, logic, software verification, temporal logic.*

I. INTRODUCTION

A basic hybrid dynamical system is one whose state may either evolve continuously for some duration of time according to one set of differential equations or be abruptly reset to a new value from which evolution is governed by another set of differential equations, with the switches typically triggered by the occurrence of some discrete event. The coordinate variables of the state may take their values in the real numbers or in a discrete (usually finite) set. The hybrid phenomena captured by such mathematical models is manifested in a great diversity of complex engineering applications, including air-traffic control, automotive control, robotics, automated manufacturing, and chemical process control, as illustrated in companion papers in this special issue. The last decade has seen considerable research effort in both computer science and control theory directed at the study of mixed discrete and continuous systems [1]–[10]. In particular, the high-confidence and safety-critical nature of the application areas has fostered a large and growing body of work on *formal methods* for hybrid systems: mathematical logics, computational models and methods, and computer-aided reasoning tools supporting the formal specification and verification of performance requirements for hybrid systems, and the design and synthesis of control structures for hybrid systems that are provably correct with respect to formal specifications. Broadly stated, formal methods are a means to *mathematicize*, and thence to *mechanize*, or render computational, what it means for a system design to "*get it right*": to correctly implement or satisfy precisely stated, unambiguous performance specifications. This paper offers a tutorial survey and a fresh perspective on the use of logics and formal methods in the analysis and synthesis of hybrid control systems.

Manuscript received October 18, 1999; revised March 14, 2000. This work was supported by U.S. ARO under Grant DAA H04-96-1-0341. The work of J. M. Davoren was supported by U.S. ONR under Grant N 00014-98-1-0535.

J. M. Davoren is with the Computer Sciences Laboratory, Research School of Information Sciences and Engineering, Australian National University, Canberra, Australia (e-mail: j.m.davoren@anu.edu.au).

A. Nerode is with the Department of Mathematics, Cornell University, Ithaca, NY 14853 USA (e-mail: anil@math.cornell.edu).

Publisher Item Identifier S 0018-9219(00)06456-2.

A. Overview: Logics and Formal Methods for Hybrid Systems

The theory and practice of formal methods in the analysis of computer hardware and software is well established. The field has been active for over 30 years, and has more recently enjoyed some industrial and commercial success; the recent survey paper [11] gives an overview. Hardware systems and software programs are traditionally modeled as purely *discrete* systems: state variables take their values in discrete (finite or countable) sets, and state transitions are modeled as occurring in a discrete, step-wise fashion. The elementary system model is that of a *finite-state automaton*, the mathematics of which forms the core theory of computer science. Within the discrete realm, these sequential state machines have been enriched in many and various ways to incorporate features of reactive, concurrent, and distributed computer systems. In the move to real-time and hybrid systems, researchers in the computer science tradition have similarly sought to extend formal methods by enriching their system

models and formal logics to deal with real-valued state variables and state transitions that model evolution according to differential equations.

Formal methods for the analysis of discrete systems fall roughly into three overlapping camps, which have carried over to hybrid discrete+continuous systems:

- *logic-based* approaches [12]–[23];
- *automata-theoretic* approaches [18], [20], [24]–[28];
- *process algebra* approaches [28], [29];

with the reference lists intended as representative samples. Our focus is on logic-based approaches, although in the course of this paper, we will briefly discuss and give some pointers to the other two approaches and note some inter-relationships between the three.

Given the tutorial nature of this paper and the breadth of its intended audience, we adopt the pedagogical course of making a first pass through the key conceptual and technical ingredients, in several introductory subsections, with a view to equipping the reader with a big picture overview of the enterprise, before embarking on the detailed technical development in the body of this paper.

In structuring our exposition, we draw on a paradigm framework for logic-based formal methods set out in the influential work of Manna and Pnueli in [12]–[14] and [30] and widely used in the field; the functional parts of the framework are illustrated in the lower gray box in Fig. 1. In [13], the framework is applied first to discrete reactive systems, then to real-time extensions of reactive systems, and finally to a class of hybrid systems. Each system in the classes under consideration is formally represented as some form of *transition system model* $\mathfrak{M}$, which is a generalization of a finite automaton, and behavioral specifications are formally encoded by formulas φ of a temporal logic extending the logic linear temporal logic (**LTL**). The formal mathematical semantics of these so-called *linear* or sequence-based temporal logics are such that a formula φ is *true* in $\mathfrak{M}$, written $\mathfrak{M} \models \varphi$, exactly when every execution sequence or trajectory of the system represented by $\mathfrak{M}$ has the property encoded by φ. The logic **LTL**, along with the so-called *branching* or state-based temporal logics such as computation tree logic (**CTL**) or (**CTL***), are discussed in this special issue [31]. The syntactic primitives and the corresponding semantic constructs of the latter temporal logics allow one to express behavioral properties of *some* execution sequences, as well as *all* execution sequences, starting from a state, and the formal semantics are such that $\mathfrak{M} \models \varphi$ means every *state* in $\mathfrak{M}$ satisfies the property expressed by φ.

In this paper, following [23] and [32], we will look to the larger family of *modal logics* [33]–[35], which includes all the standard temporal logics, and in particular, to the richly expressive "parent logic" called the *propositional modal μ-calculus* (**L**μ) [33], [36], [37]. The μ-calculus is well known in the hybrid system literature, notably from the work of Henzinger and coworkers [20], [21], [38]. In earlier work on the control theory of purely discrete systems, it was essentially

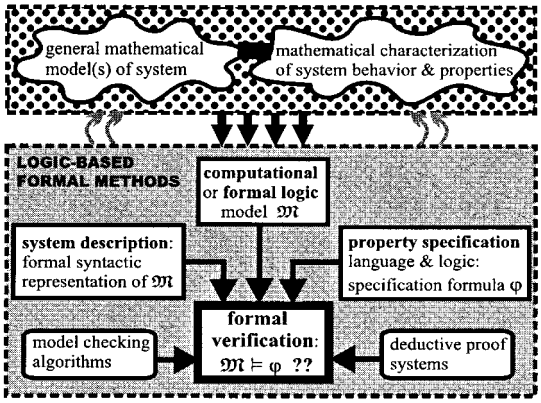


Fig. 1. Paradigm framework for logic-based formal methods.

rediscovered under the name *modular feedback logic* by Ramadge and Wonham in [39] as a formalism for stating and solving supervisory control problems for *discrete event systems* (DESs) [40].

We return to an introductory discussion of the logics and *computational* approaches to determining whether $\mathfrak{M} \models \varphi$ a little later. At this stage, the essential point is that a system and its properties are *formally represented* as models and formulas of a mathematical logic. To be able to demonstrate the degree of *faithfulness* of such formal representations, we must first develop "preformal" mathematical models of hybrid systems, and then identify and characterize in natural mathematical language both the trajectories of such systems and the sorts of properties we would like to reason formally about. This elementary point is illustrated in the upper part of Fig. 1. Note this is a separate issue from whether a particular mathematical model is an adequate representation of the concrete physical system it is intended to model. The latter issue is addressed in several branches of control theory, including studies of *system identification*, and studies of *robustness*. We return to robustness issues later in this paper; the idea there is that one has a *nominal* model of a system together with an *uncertainty class* characterizing how the *true* model might differ from the nominal one [25].

B. Overview: Mathematical Models

As our basic mathematical model, we take a class of systems known as *hybrid automata*, which have gained wide acceptance since their introduction in [18] and [19]. The same model or generalizations of it are used in several other papers in this special issue [31], [41]–[43], and the *switched systems* considered in [44] are close relatives. A (basic) hybrid automaton is a closed system with a "built-in" control structure determining when and how the system switches between its various discrete *modes*, where the continuous behavior in each discrete mode is governed by a vector differential equation (or differential inclusion).

In contrast, the supervisory control perspective on hybrid systems retains a clear separation between *plant* and *control*; the theory is developed in this special issue [45] and adapts DES control theory to the hybrid setting. A *hybrid control*

system consists of a finite control automaton operating in a closed feedback loop with a continuous plant, with communication via AD and DA interface maps. This quintessentially hybrid configuration is closely related to the switching controller architecture in [41] and is the focus of earlier work by the second author [46], among many others. In Section II, we show how the resulting closed-loop system gives rise to a basic hybrid automaton, and is thus amenable to logic-based formal specification and analysis. As an illustrative example, to which we return throughout the text, we consider the hybrid control of a simple double-integrator plant in $\mathbb{R}^2$; the example is well known in the DES-based hybrid systems literature and appears again in [45]. In the class of control problems we examine, the task is to construct a hybrid control system *so that* the associated hybrid automaton satisfies a prioritized list of performance specifications, the first of which is *safety* property.

Safety or *invariance* properties have gained the most attention in studies of hybrid systems. These are properties of the form: "All state trajectories of a system H starting from a set $Init$ of initial states remain in a prescribed set P at all times;" equivalently, in terms of *reachability*: "The set of states H-reachable from $Init$ is contained in P." For example, in an air-traffic control example, the "good" set P could be the set of states in which the distance between any pair of aircraft is greater than some minimum separation value [42], [43]. Given this relation between safety and reachability, a good deal of research effort has focused directly on techniques for either computing exactly, or else approximating, the *reachable regions* for various classes of hybrid systems, with diverse approaches drawn from optimal control, game theory, and computational geometry; see, for example, [41] and [42]. Other properties investigated and formalized include qualitative temporal notions of liveness (non-Zenoness, and switching modes infinitely often), deadlock freedom, eventuality, and fairness along infinite trajectories; qualitative ordering of events along trajectories; and quantitative timing properties of hybrid or real-time trajectories [12], [13], [15], [16], [20], [21].

From the perspective of control and systems theory, the classical concerns center on notions of *stability* and of *robustness* of systems. For example, one basic notion of *stability* is the property: "For every $\varepsilon > 0$, there is a $\delta > 0$ such that every H-trajectory that starts within distance δ from an H-invariant set P always remains within ε of P." While a variety of mathematical formulations of these concepts have been proposed for hybrid and switched dynamical systems (stability is surveyed in this issue in [44]), there has been little work to date on integrating these concerns within a framework for formal methods [25], [47]. There is perhaps good reason for this. Coming as they do from computer science, formal methods traditionally lie in the realm of *discrete mathematics*, while these notions from control theory lie squarely in the realm of *continuous mathematics*. Well before hybrid systems, the classic systems theory text of Kalman *et al.* [48] sought commonality between the two competing realms. In a chapter entitled "Automata theory: the rapprochement with control theory" (written by Arbib), we find:

> "One thing an automata theorist must often envy a control theorist is the use of continuity" (p. 179).

For hybrid automata theorists, it should go beyond envy. Developing ideas in [23], [46] and [49]–[51], we argue that a common ground is to be found by adopting the language and viewpoint of *general topology*, and that natural and imposed topological and metric structure on the state spaces of system models, and concepts of continuity with respect to such topologies, can and should be reflected in one's formal models and logics.

C. Overview: Computational Models and Methods

Work on formal methods for hybrid and real-time systems has produced a large variety of formal or computational models. The common core is to be found in a very simple class of structures called *labeled transition systems* or *LTS models* for short. An LTS model $\mathfrak{M}$ is an abstract structure consisting simply of a *state space* S of arbitrary cardinality; a collection of *binary relations* $\xrightarrow{a} \subseteq S \times S$; and a collection of distinguished *sets* $P \subseteq S$ of states [33], [52].

An LTS model is best viewed as an *abstract dynamical system*, or an *abstract machine*, whose mathematical structure is uniform across the discrete-continuous divide. When S is a finite set and the collections of relations (corresponding to an input alphabet) and of distinguished subsets (output alphabet) are both finite, such an $\mathfrak{M}$ is just a notational variant of a nondeterministic *finite automaton*. In representing a basic hybrid automaton H as an LTS model $\mathfrak{M}_H$, the system state space is an uncountable set $S := Q \times X$ where Q is a finite set of control modes and $X \subseteq \mathbb{R}^n$. One of the key insights in the hybrid systems literature, dating back to [12], [13], and [18] and earlier work on (real-) *timed automata* [53], is that both sorts of system dynamics—continuous evolution according to differential equations and discrete switches or resets of state—can be uniformly and faithfully represented as binary transition relations over a hybrid state space. The distinguished state sets include initial states, target or avoidance regions, and structural components of the hybrid automaton.

Computational or algorithmic problems in formal verification take the form:

> **Generic schema for logic-based formal verification or analysis problems:**
> Given a formal $model$ $\mathfrak{M}$ of a system design,
> together with a specification formula φ
> encoding a system property,
> the task is to $determine\ whether$ $\mathfrak{M} \models \varphi$,
> and if not, produce a counter-example
> witnessing how $\mathfrak{M}$ $fails$ to satisfy φ.

For a demarcated class of formal models and class of specification formulas, an algorithmic solution is a "black-box" computer program, which takes as input a pair $(\mathfrak{M}, \varphi)$ in the given class and returns as output either the answer YES, preferably with a transcript of the steps taken to arrive at this

answer, or else a concrete counter-example to φ extracted from the model $\mathfrak{M}$.

The computational focus brings out the issue of the *formal description* of models, as identified in Fig. 1. In order for it to be *data* for a computer program, the components of the formal model $\mathfrak{M}$, and the underlying system model out of which $\mathfrak{M}$ is formed, must be precisely described in the syntax of some formalism, such as a programming language, a graphical formalism like statecharts, or a more general-purpose formalism like first-order logic.

Some further introductory discussion of mathematical logics is in order. *First-order logic* or *predicate logic* [34] is just the logic that is used informally in the language of everyday mathematics. For example, the standard definition of a function $f\colon \mathbb{R} \to \mathbb{R}$ being continuous at a point $x \in \mathbb{R}$ with respect to a metric d on $\mathbb{R}$ is written in "informal" first-order logic, with variables ranging over the real numbers, as

$$(\forall \varepsilon > 0)(\exists \delta > 0)(\forall y)\colon d(x, y) < \delta \to d(f(x), f(y)) < \varepsilon.$$

This is "informal" since we use $(\forall \varepsilon > 0)\cdots$ as an abbreviation for $(\forall \varepsilon)$: $(\varepsilon > 0) \to \cdots$ and $(\exists \delta > 0)\cdots$ for $(\exists \delta)$: $(\delta > 0) \wedge \cdots$, and the terms $f(x)$ and $d(x, y)$ would formally be expanded to expressions built from the primitive function and constant symbols of the first-order language in use. For example, $f(x)$ could be $2x^3 - \exp(x^2) + 17$ if the language $\mathcal{L} = \{<, +, -, \cdot, \exp, 0, 1\}$ contained these function symbols, plus constants for the integers and the relation symbol $<$. In the formula above, the variable x is not bound by an $\exists$ or $\forall$ quantifier, so it is called a *free variable*. Writing $\theta(x)$ for that formula, the set-theoretical expression $\{x \in \mathbb{R} \,|\, \theta(x)\}$ means the subset of all points in $\mathbb{R}$ at which f is continuous with respect to d, and $\theta(x)$ is said to *define* this set.

A low-level formal description of an LTS model of a basic hybrid automaton consists of a finite list of first-order formulas: formulas $\theta_P(q, x_1, \cdots, x_n)$ defining the state sets $P \subseteq S \subseteq Q \times \mathbb{R}^n$, where the variable q ranges over the finite set Q of discrete states and the variables x_i range over $\mathbb{R}$ (technically, this is *multisorted* or *typed* first-order logic), and formulas $\rho_a(q, x_1, \cdots, x_n, q', x'_1, \cdots, x'_n)$ with two discrete variables and $2n$ real-valued variables, defining the relations $\xrightarrow{a} \subseteq S \times S$. The semantics of high-level hybrid programming languages such as SHIFT [54] and `Hybrid cc` [55] can be given in terms of hybrid automata, and so admit a low-level formal description of this kind.

Modal and temporal logics are best viewed as fundamentally *second-order* logics for reasoning about *sets of states*, and *operations* on *sets of states*, as distinct from first-order logics in which one reasons about elements in the domain of interpretation and functions and predicates of elements.

The formal semantics of the μ-calculus, and its (state-based) modal and temporal sublogics, define the meaning or *denotation set* $[\![\varphi]\!]^{\mathfrak{M}} \subseteq S$ of a formula φ in a model $\mathfrak{M}$. Formulas are built up starting from *propositional constants* p that name the state sets in a model, and compounds are formed using the standard logical connectives or Boolean operations $\neg$ ("*not*"), $\wedge$ ("*and*"), $\vee$ ("*or*"), $\to$ ("*if...then...*") and $\leftrightarrow$ ("*iff*"), together with various *modal* or *temporal operators*, which reflect the effect of state transitions according to the relations of a model. Among the basic modal operators are the (relativized) *box* and *diamond* operators. For relations a, a formula $[a]\varphi$ reads "*All* a-successors satisfy φ" or "a-actions *necessarily* bring about φ," while $\langle a\rangle\varphi$ reads "*Some* a-successor satisfies φ" or "a-actions can *possibly* bring about φ." Intuitively, $[\![\varphi]\!]^{\mathfrak{M}}$ is the set of states that satisfy φ in $\mathfrak{M}$, and $\mathfrak{M} \models \varphi$ exactly when $[\![\varphi]\!]^{\mathfrak{M}} = S$.

The two main methods for the verification of modal or temporal logic properties are *model checking algorithms* and *deductive proof systems*. The essential task of a model checking algorithm is to recursively *compute* the denotation set $[\![\varphi]\!]^{\mathfrak{M}}$, and if the complement $S - [\![\varphi]\!]^{\mathfrak{M}} = [\![\neg\varphi]\!]^{\mathfrak{M}}$ is nonempty, these states provide the required counter-examples. For hybrid systems, this necessarily falls under the heading *symbolic* model checking, since over an infinite state space, one needs a finitary syntactic or symbolic means of representing sets of states and operations on them; over a finite state space, one can resort to explicit enumeration. When the component state sets and relations of a model $\mathfrak{M}$ are formally defined in first-order logic, one can seek to use the same representation (and in particular, *quantifier-free* first-order formulas) in the course of model checking. As examined in [31] in this special issue, the *decidability*, or possibility of an algorithmic solution guaranteed to terminate on all inputs, for model checking of temporal logic properties for various classes of hybrid automata, depends crucially on the syntactic complexity and form of the first-order formulas defining the components of the systems.

The other approach to formal verification is deductive, and there are usually several different types of *deductive proof systems* that can be developed for a logic. The simplest is called a *Hilbert-style* or *axiomatic* proof system, which consists of a collection of formulas designated as *axioms*, and a collection of *inference rules* of the form

$$\frac{\psi_1\, \psi_2\, \cdots\, \psi_m}{\varphi}.$$

A Hilbert-style proof system $\mathcal{S}$ is said to be *sound* with respect to a class of models $\mathcal{K}$ if for each $\mathfrak{M}$ in $\mathcal{K}$, each of the axioms of $\mathcal{S}$ is true in $\mathfrak{M}$ and whenever all of the premises of an inference rule in $\mathcal{S}$ are true in $\mathfrak{M}$, then the conclusion of that rule is also true in $\mathfrak{M}$. Deductive verification of $\mathfrak{M} \models \varphi$ starts with a list Γ of formulas ψ such that $\mathfrak{M} \models \psi$ is immediate from the formal description of $\mathfrak{M}$, or has otherwise already been established, and then seeks a formal proof, or sequence of inference steps in $\mathcal{S}$, demonstrating that φ is a *deductive consequence* of Γ. By soundness, one can then conclude $\mathfrak{M} \models \varphi$. Hilbert-style axiomatizations are important for clarifying and understanding a logic, and are easy enough to use manually, but they do not readily lend themselves to automated proof search or to the construction of counter-examples. Other types of deductive systems such as *tableaux systems* or *Gentzen-style* proof systems [34], [56], which produce labeled tree or graph-style proofs, are better suited to these tasks.

A number of the logics developed for hybrid and real-time systems consist of multisorted first-order logic *combined* with some temporal operators to give a single formalism for *both* description of system components and specification of system properties; for example, *temporal logics of actions* TLA+ and cTLA [16], [17] and *extended duration calculus* EDC [15], [28]. For these, deductive verification is the only available approach.

The bulk of the work on logics and formal methods for hybrid systems, as for discrete systems, has focused on the "after-the-fact" verification of a completed system design. We are interested in ways in which the *same* technical machinery of model checking and deductive proof systems can be used for the synthesis or construction of a system from an incomplete design.

> **Generic schema for logic-based formal synthesisproblems:**
> Given a performance specification formula φ,
> and an *incomplete* or *under-determined*
> description of a formal model,
> the task is to "fill in the blanks"
> and *construct* an $\mathfrak{M}$ *so that* $\mathfrak{M} \models \varphi$
> or else determine that no such $\mathfrak{M}$ exists.

In the example control problem we consider in Section II, the blanks to fill in are an AD map and a finite control automaton; together with the given plant model and DA map, the closed-loop system forms "a" hybrid automaton. We first describe the construction in general terms, then return to it in later sections to show how modal logics can be used not only to formalize the performance requirements, but also to formalize lower level decisions and computations required in the course of the construction; we generate a list of simpler formulas that are true by construction, and from these we can deductively derive the desired specification formulas, so establishing that the construction is correct.

For comparison, [41] in this special issue considers a class of control problems in which one starts with a complete hybrid automaton H, and the synthesis task is to find the largest *subsystem* $H' \preceq H$ such that H' satisfies a safety property. While [41] does not use a logic framework, we will briefly sketch in Section V-C how that type of construction can be formalized in the μ-calculus, and its relation to similar *maximal invariant subset* constructions in DES control theory [39].

The body of this paper is roughly structured around Fig. 1. In Section II, we examine mathematical models of hybrid systems and their elementary properties and set up our hybrid control example. Section III covers transition system models, and the formal representation of hybrid automata, plus a brief discussion of automata-theoretic and process algebra approaches to hybrid systems. The longer Section IV introduces and develops modal and temporal logics for the specification of system properties, while Section V surveys model checking and deductive proof systems, and logic-based approaches to the design and synthesis of control structures for hybrid systems. The concluding Section VI discusses related and ongoing work.

II. Mathematical Models

A. Preliminaries

For reference, we include a glossary of notation in Table I, with the right-hand column giving the subsection in which the notation is defined.

As identified in the introduction, the elementary mathematical objects of interest are *relations* or *set-valued functions*, which are the nondeterministic analog of functions. Following the useful convention in set-valued analysis [57], the notation $r\colon X \rightsquigarrow Y$ will be used to mean $r\colon X \to \mathcal{P}(Y)$ is a set-valued function, with set-values $r(x) \subseteq Y$ for each $x \in X$ (possibly $r(x) = \emptyset$), or equivalently, $r \subseteq X \times Y$ is a *relation*, sometimes called the *graph* of a set-valued function. For points $x \in X$ and $y \in Y$, the expressions $x \xrightarrow{r} y$, xry, $(x, y) \in r$, and $y \in r(x)$ are to be read as synonymous; in words, y is an r-*successor* of x, or x is an r-*predecessor* of y. The *domain* of a relation $r\colon X \rightsquigarrow Y$ is the set $\mathrm{dom}(r) \stackrel{\mathrm{def}}{=} \{x \in X | r(x) \neq \emptyset\}$. In computer science and DES theory, a relation r is said to be *enabled* at points $x \in \mathrm{dom}(r)$. Unlike single-valued functions, every relation $r\colon X \rightsquigarrow Y$ has a natural *converse* (or *inverse*) $\breve{r}\colon Y \rightsquigarrow X$, given simply by: $(y, x) \in \breve{r}$ iff $(x, y) \in r$.

Some elementary relations of interest include: the *identity function* $id_X\colon X \to X$; partial functions $test.A\colon X \rightsquigarrow X$ formed by restricting id_X to a domain $A \subseteq X$, so $x \xrightarrow{test.A} x'$ iff $x \in A$ and $x' = x$; and *set-valued constant* maps $r = A \times B$, which means $r(x) = B$ for each $x \in A = \mathrm{dom}(r)$.

The (sequential) *composition* of relations $r\colon X \rightsquigarrow Y$ and $s\colon Y \rightsquigarrow Z$ will be written $r \circ s\colon X \rightsquigarrow Z$ (abbreviated rs) in sequential (word) order, as is usual in computer science, but the reverse of the usual order for functional composition; see [57] and [58]. Composition is explicitly defined by $x \xrightarrow{r \circ s} z$ iff $(\exists y \in Y)[x \xrightarrow{r} y$ and $y \xrightarrow{s} z]$. Given relations $r\colon X \rightsquigarrow Y$ and $s\colon X \rightsquigarrow Y$, their *relational union* (*sum* or *choice*) $r \cup s\colon X \rightsquigarrow Y$ is just the union of r and s considered as subsets of $X \times Y$. For $r\colon X \rightsquigarrow X$, the k-*fold composition* r^k for $k \in \mathbb{N}$ is defined inductively by $r^0 = id_X$ and $r^{k+1} = r \circ r^k$. The *Kleene star* operation (reflexive-transitive closure) produces the relation $r^*\colon X \rightsquigarrow X$ by taking the infinite union of all the r^k for $k \in \mathbb{N}$. A *regular expression* of relations is one formed using the operations of sequential composition, finite union and Kleene star.

A (nondeterministic) *finite automaton* is a structure $\mathcal{A} = (Q, \Sigma, \Phi, \delta, \lambda)$, where $Q \neq \emptyset$ is the finite set of *states*, Σ is the finite *input alphabet*, Φ is the finite *output alphabet*, $\delta\colon Q \times \Sigma \rightsquigarrow Q$ is the *transition relation*, and $\lambda\colon Q \rightsquigarrow \Phi$ is the *output relation*. The *input–output* relation $\gamma\colon \Sigma \rightsquigarrow \Phi$ of $\mathcal{A}$ is given by $(a, p) \in \gamma$ iff $(\exists q, q') \in Q)\colon q' \in \delta(q, a)$ and $p \in \lambda(q')$. A finite automaton $\mathcal{A}$ is called *deterministic* if the transition map δ and the output map λ are both single-valued functions (but possibly only *partial* functions).

We also use some elementary notions from general topology; [59] is a useful text, and [58] and [60] develop the general topology of relations/set-valued maps. Recall that a *topology* $\mathcal{T}$ on a set S is abstractly defined as a family $\mathcal{T} \subseteq \mathcal{P}(S)$ of subsets of S that contains S and $\emptyset$ and is closed under finite intersections and arbitrary unions. The

Table 1
GLOSSARY OF NOTATION

$\mathbb{R}$	real numbers	
$\mathbb{R}^+$	non-negative reals, $[0, \infty)$	
$\mathbb{N}$	natural numbers	
$\mathcal{P}(X)$	power-set of X	
$(z_i)_{i\in I}$	infinite or finite sequence, with $I = \mathbb{N}$ or $I = \{0, 1, ..., N\}$	
$f : X \to Y$	(single-valued) function from X to Y	
$r : X \rightsquigarrow Y$	set-valued map or relation from X to Y	II.A
$test.A$	id_X restricted to $A \subseteq X$	II.A
ϕ	flow of vector field on $X \subseteq \mathbb{R}^n$	II.B
H	hybrid automaton (HA)	II.B
η	trajectory of a HA	II.B
h	H-reachability relation	II.B
$\mathfrak{M}$	LTS model	III.A
(Σ, Φ)	modal signature of LTS model	III.A
$L(\mathfrak{M})$	automata language of $\mathfrak{M}$	III.A
$e(A, \phi)$	relation of evolution along flow ϕ within A	III.B
$\mathfrak{M}_H$	LTS model of a HA H	III.B
B_δ	metric tolerance relation	III.B
φ	**PML** or **L**μ formula	IV.B
$\langle a\rangle\varphi$	*diamond-a* modal operator	IV.B
$[a]\varphi$	*box-a* modal operator	IV.B
$\mathrm{Pre}^\exists(r)$	$\exists$-pre-image operator of r	IV.B
$\mathrm{Pre}^\forall(r)$	$\forall$-pre-image operator of r	IV.B
$[\![\varphi]\!]^{\mathfrak{M}}$	denotation $[\![\varphi]\!]^{\mathfrak{M}} \subseteq S$ in $\mathfrak{M}$ of **PML** or **L**μ sentence	IV.B,E
$\mathfrak{M}, s \vDash \varphi$	φ is satisfied at state s in $\mathfrak{M}$	IV.B,E
$\mathfrak{M} \vDash \varphi$	φ is true in $\mathfrak{M}$	IV.B,E
$\mathcal{B}(\mathfrak{M})$	minimal modal algebra for LTS model $\mathfrak{M}$	IV.B
$\mu Z.\varphi$	least fixed-point quantifier	IV.E
$\nu Z.\varphi$	greatest fixed-point quantifier	IV.E
ξ	propositional or set-valued variable assignment	IV.E
$[\![\varphi]\!]^{\mathfrak{M}}_\xi$	denotation $[\![\varphi]\!]^{\mathfrak{M}}_\xi \subseteq S$ in $\mathfrak{M}$ of **L**μ formula φ under ξ	IV.E

sets U in $\mathcal{T}$ are called *open*, and their complements are called *closed*. The topological interior $int_{\mathcal{T}}(A)$ of a set $A \subseteq S$ is the largest open set contained in A, while the dual closure $cl_{\mathcal{T}}(A)$ is the smallest closed set containing A. Sets $X \subseteq \mathbb{R}^n$ will be equipped with the standard Euclidean metric (subspace) topology unless otherwise indicated.

B. Basic Hybrid Automata and Their Trajectories

We take the standard ingredients of the popular hybrid automaton model [18]–[20], [31], but reformulate them slightly, keeping in clear view their subsequent representation in a transition system model.

A continuous dynamical system $\dot{\mathbf{x}}(t) = F(\mathbf{x}(t))$ on a set $X \subseteq \mathbb{R}^n$ given by a Lipschitz continuous vector field $F\colon X \to \mathbb{R}^n$ has, for each initial state $\mathbf{x} \in X$, a unique solution (or *integral curve*) $\gamma_{\mathbf{x}}\colon I \to X$ where $I \subseteq \mathbb{R}$ contains 0, $(d/dt)\gamma_{\mathbf{x}}(t) = F(\gamma_{\mathbf{x}}(t))$ for all $t \in I$, and $\gamma_{\mathbf{x}}(0) = \mathbf{x}$. Under additional assumptions—for example, X is compact and F is continuously differentiable—the time domain of the solutions may be extended to all of $\mathbb{R}$, and the system has a *global flow* $\phi\colon X \times \mathbb{R} \to X$ with $\phi(\mathbf{x}, t) = \gamma_{\mathbf{x}}(t)$ [61]. For our purposes, it suffices to know that a flow ϕ is continuous in both arguments separately and satisfies the *flow laws*: $\phi(\mathbf{x}, 0) = \mathbf{x}$ and $\phi(\mathbf{x}, t + s) = \phi(\phi(\mathbf{x}, t), s)$ for all $\mathbf{x} \in X$ and $t, s \in \mathbb{R}$, i.e., it respects $\mathbb{R}$ as an additive group. A *semiflow* $\phi\colon X \times \mathbb{R}^+ \to X$ is just a flow defined on the nonnegative time axis.

A more general class of systems is obtained by allowing the continuous dynamics to be governed by a *differential inclusion* $\dot{\mathbf{x}}(t) \in F(\mathbf{x}(t))$, where $F\colon X \rightsquigarrow \mathbb{R}^n$ is a set-valued vector field. As examined in the companion paper [31], there is particular interest in dynamics of the form $\dot{\mathbf{x}}(t) \in B$, where $B = [a_1, b_1] \times \cdots \times [a_n, b_n]$ is a *rectangle* or *box* in $\mathbb{R}^n$, so a_k and b_k are fixed lower and upper bounds on the rate of change in the coordinate x_k. The associated *set-valued flow* $\phi\colon X \times \mathbb{R} \rightsquigarrow X$ is then given by the formula $\phi(\mathbf{x}, t) = \{\mathbf{x} + t\mathbf{u} \,|\, \mathbf{u} \in B\}$. While the framework developed here can be adapted to deal with differential inclusions, we focus on the simpler and conceptually clearer basic case.

Definition 1: A (basic) *hybrid automaton* (HA) is a system

$$H = (Q, E, X, \{\phi_q, Inv_q\}_{q\in Q}, \{R_{q,q'}, Grd_{q,q'}\}_{(q,q')\in E})$$

where:

- Q is a finite set of *discrete states*, also called *control modes* or *control locations*;
- $E \subseteq Q \times Q$ is the *discrete transition relation* or *control graph*;
- $X \subseteq \mathbb{R}^n$ is the *continuous state space*, the valuation space for a vector of n real-valued variables;
- for each discrete control mode $q \in Q$:
  - $\phi_q\colon X \times \mathbb{R} \to X$ is a *flow* on X, giving the continuous dynamics in mode q;
  - $Inv_q \subseteq X$ is the set of *invariant* states for mode q, or the *domain of permitted evolution* within mode q;
- for each discrete transition pair $(q, q') \in E$:
  - $R_{q,q'}\colon X \rightsquigarrow X$ is a *reset relation*, defining the possible successors $\mathbf{x}' \in R_{q,q'}(\mathbf{x})$ of a point $\mathbf{x} \in X$ upon switching from q to q';
  - $Grd_{q,q'} \stackrel{\text{def}}{=} \mathrm{dom}(R_{q,q'}) \subseteq X$ is the *guard region* or *enabling event* for a switch from q to q'.

The *state space* is $S_H \stackrel{\text{def}}{=} Q \times X$. The subset of *admissible states* is $Inv_H \stackrel{\text{def}}{=} \cup_{q\in Q}(\{q\} \times Inv_q)$. A set of *initial states* $Init \subseteq Inv_H$ may also be given. HA are usually represented graphically as in Fig. 2.

The system is permitted to evolve according to ϕ_q only while the state is in Inv_q. The sets Inv_q can arise from physical modeling considerations or decisions in system design. In general, the reset relations can be arbitrary set-valued maps, and are operationally thought of as nondeterministic assignments triggered by the event of reaching a guard set. Particular reset relations of interest include the restriction to $Grd_{q,q'}$ of the identity function (i.e., the partial function $test.Grd_{q,q'}$) [41] or the restriction to $Grd_{q,q'}$ of a function that is the identity on some real-valued coordinates and set-valued constant on the remainder [31].

Definition 2: A *trajectory* of a hybrid automaton H is a finite or infinite sequence $\eta = \langle \Delta_i, q_i, \gamma_i\rangle_{i\in I}$ such that for each $i \in I$:

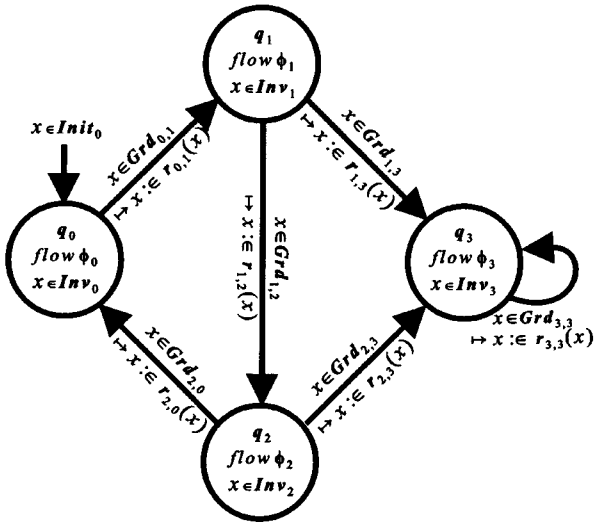


Fig. 2. Graphical representation of a basic hybrid automaton.

- the duration $\Delta_i \in \mathbb{R}^+ \cup \{\infty\}$, with $\Delta_i = \infty$ only if I is finite and $i = \max(I)$;
- the discrete state $q_i \in Q$;
- the curve $\gamma_i : [0, \Delta_i] \to X$ is such that for all $t \in [0, \Delta_i]$, $\gamma_i(t) = \phi_{q_i}(\gamma_i(0), t)$ and $\gamma_i(t) \in Inv_{q_i}$; i.e., γ_i is a continuous curve along the flow ϕ_{q_i} that lies entirely inside Inv_{q_i}, with interval $[0, \infty)$ if $\Delta_i = \infty$;
- if $i < \max(I)$, then $(q_i, q_{i+1}) \in E$ and the adjacent end-points satisfy $\gamma_i(\Delta_i) \overset{R_{q_i,q_{i+1}}}{\longrightarrow} \gamma_{i+1}(0)$.

The *cumulative duration* $\Delta(\eta)$ of a trajectory η is the sum $\Delta(\eta) \overset{\text{def}}{=} \sum_{i \in I} \Delta_i$. A H-trajectory η will be called *jump-finite* (*jump-infinite*) if I is finite (infinite); *time-finite* (*time-infinite*) if $\Delta(\eta) < \infty (\Delta(\eta) = \infty)$; and *finite* if it is so in both senses.

The *H-reachability relation* h: $S_H \rightsquigarrow S_H$ is defined by

$$(q, \mathbf{x}) \xrightarrow{h} (q', \mathbf{x}')$$
$$\overset{\text{def}}{\Leftrightarrow} \text{there is a finite } H\text{-trajectory } \eta = \langle \Delta_i, q_i, \gamma_i \rangle_{0 \le i \le N}$$
$$\text{such that } q = q_0 \text{and } \mathbf{x} = \gamma_0(0)$$
$$\text{and } q' = q_N \text{ and } \mathbf{x}' = \gamma_N(\Delta_N).$$

In the reminder of this subsection, we develop just enough of the mathematics of this class of systems to ground our subsequent work in stating and solving a control problem, and formalizing system behavior and properties in formal model and logics.

The *time line* along a hybrid trajectory is a lexicographically ordered subset of $\mathbb{N} \times \mathbb{R}^+$, and a *time position* is a pair (i, t) where i is the step number and $t \in [0, \Delta_i]$; the *state* of η at time position (i, t) is $(q_i, \gamma_i(t))$, and the *cumulative time* at position (i, t) is $t + \sum_{k<i} \Delta_k$. Note that the reset actions are assumed to occur *instantaneously*, between the time positions (i, Δ_i) and $(i + 1, 0)$, which have the same cumulative time.

For each $i < \max(I)$ in an H-trajectory η, the definition entails that $\gamma_i(\Delta_i) \in Grd_{q_i,q_{i+1}} \cap Inv_{q_i}$ and $\gamma_{i+1}(0) \in Inv_{q_{i+1}}$. This means that if a flow ϕ_q can leave Inv_q without first reaching a set $Grd_{q,q'}$, any trajectory with that behavior will hit a *dead end* or become *blocked* at the topological boundary of Inv_q. Also, if $R_{q,q'}(\mathbf{x}) \not\subseteq Inv_{q'}$ for some $\mathbf{x} \in Grd_{q,q'} \cap Inv_q$, then trajectories reaching $(q, \mathbf{x})$ can also be blocked.

A finite H-trajectory $\eta = \langle \Delta_i, q_i, \gamma_i \rangle_{0 \le i \le N}$ may be *concatenated* with another (arbitrary) H-trajectory $\eta' = \langle \Delta'_j, q'_j, \gamma'_j \rangle_{j \in J}$, with the resulting trajectory written $\eta * \eta'$, provided $q'_0 = q_N$ and $\gamma'_0(0) = \gamma_N(\Delta_N)$. The collection of all H-trajectories is partially ordered by the *prefix* or *extension* relation $\sqsubseteq$, defined by $\eta \sqsubseteq \eta'$ iff either $\eta = \eta'$ or there is a H-trajectory η'' such that $\eta' = \eta * \eta''$. An H-trajectory is *maximal* with respect to $\sqsubseteq$ iff either it is jump-infinite, or it is jump-finite and time-infinite, with the flow ϕ_{q_N} from $\gamma_N(0)$ remaining invariantly in Inv_{q_N}, or else it is finite with the last state $(q_N, \gamma_N(\Delta_N))$ a blocked state.

A *jump-infinite liveness* property of a HA is the condition that every maximal H-trajectory starting from a given set $Init$ is jump-infinite. A distinct liveness property is the *non-Zeno* condition: there are no H-trajectories that are jump-infinite but time-finite. Such trajectories are mathematically possible but not physically realizable; the Zeno phenomena is a manifestation of *chattering* in classical control theory. A simple sufficient (but not necessary) condition for a trajectory $\eta = \langle \Delta_i, q_i, \gamma_i \rangle_{i \in I}$ to be non-Zeno is the existence of a $d > 0$ such that $\Delta_i \ge d$ for all $i \in I$.

For a state $(q, \mathbf{x}) \in S_H$, the set $h(q, \mathbf{x})$ is the collection of all states $(q', \mathbf{x}')$ that are *reached* by some H-trajectory starting from $(q, \mathbf{x})$, and the domain $\text{dom}(h) = Inv_H$ is just the admissible states. For any set $A \subseteq S_H$, the *H-reachable* region *from* A is the direct image $h(A) = \bigcup \{h(q, \mathbf{x}) | (q, \mathbf{x}) \in A\}$. When $Init \subseteq Inv_H$ is given, the set $h(Init)$ is often referred to as $\text{Reach}(H)$, *the reachable region* of H. A set $A \subseteq S_H$ is *H-invariant* if $s \in A$ implies $h(s) \subseteq A$; equivalently, every H-trajectory that starts in A always remains within A. In particular, postimage sets $h(A)$ (and $\text{Reach}(H)$) are immediately H-invariant.

C. Closed-Loop Hybrid Control and Hybrid Automata

A quintessentially *hybrid* control configuration is a finite control automaton operating in a closed feedback loop with a continuous plant, depicted in Fig. 3. This is the focus of a DES approach to the control of hybrid systems, developed in detail in this issue in [45]. Our purpose here is to show how such hybrid closed-loop systems *give rise* to basic hybrid automata and to lay the groundwork for the formulation of a class of synthesis problems.

Under the DES approach, the finite control automaton is usually taken to be *deterministic*, and both the AD-interface map α: $X \to P$ ("generator") and the DA-interface map β: $C \to U$ ("actuator") are total, single-valued *functions*. On the DA side, a control action symbol $c \in C$ is mapped to a single constant input vector $\beta(c) \in U \subseteq \mathbb{R}^m$, which is fed into the plant equation $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$ for use until the next control switch. The function β thus determines a finite family of flows ϕ_c: $X \times \mathbb{R} \to X$ indexed by $c \in C$. On the AD side, the function α determines a finite *partition*

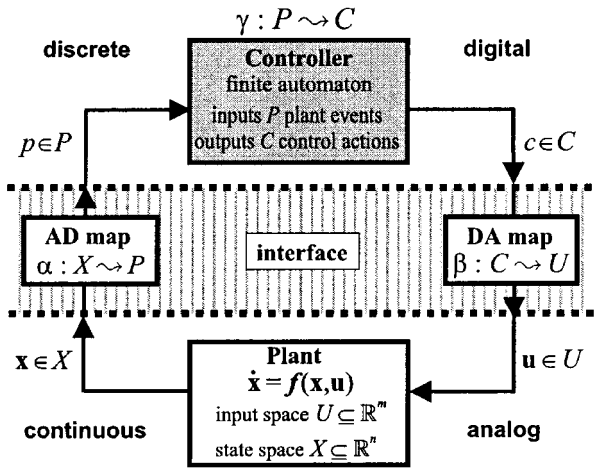


Fig. 3. Closed-loop hybrid control system.

of the plant state space $X \subseteq \mathbb{R}^n$, with equivalence classes $A_p = \{\mathbf{x} \in X | \alpha(\mathbf{x}) = p\}$ indexed by plant events $p \in P$. The discrete plant event signal instantaneously changes from p to p', and the new p' sent as input to the control automaton, when the plant state hits (or crosses) the *common boundary* $bd(A_p) \cap bd(A_{p'})$ of the disjoint plant event regions A_p and $A_{p'}$. (Recall the topological boundary operator satisfies $bd(A) = cl(A) - int(A)$.)

In [45], a chief object of study is the finite *DES plant automaton*, formed by taking the plant together with the AD and DA interface maps, with a view to adapting and applying the Ramadge–Wonham DES theory to the purely discrete closed-loop system of DES controller and plant. Their formulation also differs slightly in that it associates plant events with $n-1$ dimensional *hypersurfaces*, which separate X into disjoint regions; any finite collection of hypersurfaces will define a partition of X, with the common boundary of any two equivalence classes' lying in one of the hypersurfaces. The definition in [45] of the closed-loop trajectories on X of their hybrid control systems is also more involved in that it allows a fixed *time delay* between the time a new plant event symbol is sent as input to the controller and the time the controller outputs a new control action to the plant. We ignore that complication for now but return to it briefly in Section III-B.

Proposition 3: Given a hybrid control loop L as in Fig. 3, with single-valued AD and DA interface maps, there is a hybrid automaton H such that the trajectories of H are in one–one correspondence with the closed-loop trajectories of L. ■

In general, there will be many such hybrid automata H. For the simplest such system, suppose all we know about the finite control automaton is its input–output relation γ: $P \rightsquigarrow C$ (which in general is set-valued even when the automaton is deterministic). A pair $(p, c) \in \gamma$ can be read as a basic instruction of the controller: in the plant region p, apply control action c. So an obvious choice for the set Q of *control modes* of the hybrid automaton is $Q: \ = \gamma \subseteq P \times C$; for each $q = (p, c) \in Q$, take $\phi_q: \ = \phi_c$, and take $Inv_q := cl(A_p)$ as the mode-invariant. Edges $(q, q') = ((p, c), (p', c')) \in E$ in the hybrid automaton control graph encode the (zero-delay) switching behavior of closed-loop trajectories by taking $Grd_{q,q'} := bd(A_p) \cap bd(A_{p'})$ and taking the reset relation as $R_{q,q'} := test.Grd_{q,q'}$.

The two interface maps α and β determine *discretizations* of the plant state space $X \subseteq \mathbb{R}^n$ and the plant input space $U \subseteq \mathbb{R}^m$, respectively. A richer class of discretizations is obtained by allowing *set-valued* interface maps, as indicated in Fig. 3. A set-valued DA map β: $C \rightsquigarrow U$ associates with each control action symbol $c \in C$ a *set* of input vectors $\beta(c) \subseteq U$; the plant is then governed by the differential inclusion $\dot{\mathbf{x}} \in F_c(\mathbf{x})$, where $F_c(\mathbf{x}) = \{f(\mathbf{x}, \mathbf{u}) \,|\, \mathbf{u} \in \beta(c)\}$, as in the generalized hybrid automata considered in [31].

Developing ideas in [46], our interest here is more in the other side. A set-valued AD map α: $X \rightsquigarrow P$ determines a family of *possibly overlapping* plant event regions $A_p = \{\mathbf{x} \in X | p \in \alpha(\mathbf{x})\} = \breve{\alpha}(p)$ indexed by $p \in P$. When α is total [$\mathrm{dom}(\alpha) = X$], such a family defines a *finite cover* $X = \bigcup_{p \in P} A_p$, and conversely, any finite cover of X determines a total AD map α: $X \rightsquigarrow P$. In defining what we mean by the closed-loop trajectories of a system with a set-valued AD map, the simplest way to do so is directly in terms of an associated hybrid automaton, so that the extension of Proposition 3 to the set-valued case will become true by definition. Assume the cover is nondegenerate, so $A_p \nsubseteq A_{p'}$ for $p' \neq p$. Then as before, let H be the hybrid automaton in which $Q := \gamma \subseteq P \times C$, but now simply take $Inv_q := A_p$ and $\phi_q := \phi_c$ for $q = (p, c) \in Q$. If it is possible to evolve under action c from a plant state in $A_p - A_{p'}$ into a state in $A_p \cap A_{p'} \neq \emptyset$, so causing a change in the set of symbols sent by α to the controller, and if $(p', c') \in Q$, then put an edge $(q, q') = ((p, c), (p', c')) \in E$, and take $Grd_{q,q'} := A_p \cap A_{p'}$ and $R_{q,q'} := test.Grd_{q,q'}$. Coming full circle, the hybrid automaton representation can be used to produce a *realization* of a finite control automaton with the given I/O relation $Q = \gamma$ by taking Q itself as the internal states, defining δ: $Q \times P \rightsquigarrow Q$ by $(p'', c'') \in \delta((p, c), p')$ iff $p'' = p'$ and $((p, c), (p'', c'')) \in E$, and taking λ: $Q \rightsquigarrow C$ as projection onto C.

The focus in [46], followed up in [62], is on the *finite topology* generated from a finite cover $\{A_p\}_{p \in P}$ by taking all (finite) unions and intersections. In particular, when each of the cover sets $A_p \subseteq X$ is open in the standard topology on $X \subseteq \mathbb{R}^n$, the resulting finite topology is a *subtopology* of the standard topology on X. We briefly return to a discussion of finite topologies in Section IV-H.

For synthesis, our interest is in general recipes for *building* hybrid automata, from the ground up, *so that* the resulting system is guaranteed to satisfy a list of performance specifications. We consider the following class of problems.

Problem 4: Given $X \subseteq \mathbb{R}^n$, $U \subseteq \mathbb{R}^m$ and a plant model $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$, together with a finite control action alphabet C and a DA map β: $C \rightsquigarrow U$, the task is to *complete the control loop* by constructing a finite plant event alphabet P, an AD map α: $X \rightsquigarrow P$, and a controller I/O relation $Q = \gamma$: $P \rightsquigarrow C$, so that the associated hybrid automaton H satisfies a prioritized list of performance specifications of the following form.

1) *Safety*: Given a proscribed set $Bad \subseteq X$, no H-trajectory shall ever enter the set Bad; the construction must produce a set $Good \subseteq X - Bad$ that is invariant under H-trajectories, which will be taken as a set of initial states.
2) *Event sequence behavior*: Given a finite collection of sets $A_k \subseteq X$, $k \in K$, with $A_k \cap Bad = \emptyset$, and a set-valued map $next$: $K \rightsquigarrow K$ prescribing an order of traversal through the regions A_k, every maximal H-trajectory starting in $Good$ shall be such that whenever it ever enters an A_k, it remains there *continually* and *until* it crosses into $A_{k'}$ for some $k' \in next(k)$.
3) *Liveness*: Every maximal H-trajectory starting from $Good$ shall be jump-infinite and time-infinite (hence nonZeno).

To ensure the problem is well posed, assume that each A_k, as well as Bad and the whole space X, are connected subsets of $\mathbb{R}^n$; for each $k' \in next(k)$, $A_k \cap A_{k'} \neq \emptyset$ is also connected; and there is a connected set $A_0 \supseteq Bad$ such that $\{A_k\}_{k\in K} \cup \{A_0\}$ covers X. This initial cover will then be *refined* to produce a cover $\{A_p\}_{p\in P}$ that gives an AD map.

D. Example Control Problem and Solution

We illustrate a general synthesis construction by way of a simple example. The same double integrator plant also appears in [45], with different control objectives. The plant dynamics over $X = \mathbb{R}^2$ and $U = \mathbb{R}$ are given by

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

i.e.,

$$\begin{aligned} \dot{x}_1(t) &= x_2(t) \\ \dot{x}_2(t) &= u. \end{aligned}$$

The input-parametrized flow ψ: $X \times U \times \mathbb{R} \to X$ is defined by degree 2 polynomials

$$\psi(x_1, x_2, u, t) = \left(x_1 + x_2 t + \frac{u}{2} t^2, x_2 + ut\right).$$

For the control action alphabet, we take $C = \{\mathbf{dn}, \mathbf{ac}, \mathbf{up}\}$, and take a single-valued DA map β: $C \to U$ given by $\beta(\mathbf{dn}) = -b$, $\beta(\mathbf{ac}) = 0$ and $\beta(\mathbf{up}) = b$, for some fixed control value $b \geq 2$. The resulting three flows are illustrated in Fig. 4.

Our concrete performance specifications are that the Bad region is the unit disk and the prescribed event sequence behavior is to proceed with a clockwise motion. Visually, we are steering a point $\mathbf{x} \in \mathbb{R}^2$ around the disk using sequences chosen from three primitive control actions.

To solve this problem, we start by fixing a margin δ of *metric tolerance*, with $0 < \delta < 1$, and let A_0 be the open δ-ball around the unit disk (using the Euclidean metric on $\mathbb{R}^2$), so $A_0 = \{(x_1, x_2) \in \mathbb{R}^2 | x_1^2 + x_2^2 < 1 + \delta\}$. We strengthen the safety requirement by this tolerance margin, and will look for the invariant set $Good$ within the *complement* of A_0. To encode the clockwise motion requirement, for each $k \in K := \{1, 2, 3, 4\}$, let A_k be the set of points

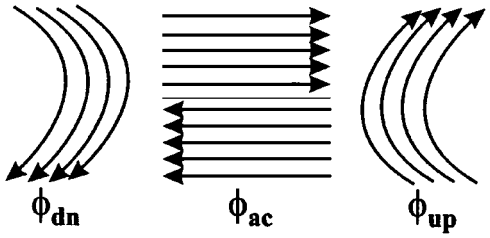


Fig. 4. Three flows of double integrator plant.

lying in the open kth quadrant of $\mathbb{R}^2$ and outside the closed unit disk, together with a δ-overlap into the $(k \ominus 1)$th quadrant, where $k \ominus 1 = k - 1$ if $k > 1$ and $k \ominus 1 = 4$ if $k = 1$. For example

$$A_2 = \left\{(x_1, x_2) \in \mathbb{R}^2 | x_1 < \delta \wedge x_2 > 0 \wedge x_1^2 + x_2^2 > 1\right\}.$$

The prescribed order of traversal is then given by $next(k) = \{k \ominus 1\}$ (single-valued), and the connected sets in $\{A_k\}_{k\in K} \cup \{A_0\}$ form an initial cover of X with δ-overlaps. The synthesis procedure then consists of three stages.

Stage 1—Refining the Initial Cover: The strategy is to identify all the subregions of A_k from which a safety violation is *possible* under one of the controls c. For each $k \in K$ and $c \in C$, let $\delta\text{-}Unsafe_{k,c}$ be the set of points in A_k that are within distance δ of some point in A_k from which c-controlled evolution can *possibly* lead *into* A_0. And let $Safe_{k,c}$ be the set of points in A_k from which c-controlled evolution *always* remains *outside* A_0. By construction, $\delta\text{-}Unsafe_{k,c}$ and $Safe_{k,c}$ overlap with metric width δ. The subregions $A_{k,m}$ of A_k then consist of the nonempty connected components of all $2^{|C|}$ possible combinations of intersections

$$(\cap_{c\in C_1} Safe_{k,c}) \cap (\cap_{c\notin C_1} \delta\text{-}Unsafe_{k,c})$$

for subsets $C_1 \subseteq C$.

Fig. 5 illustrates such a cover. For example, $A_{2,4}$ is $\delta\text{-}Unsafe_{2,\mathbf{dn}} \cap Safe_{2,\mathbf{ac}} \cap Safe_{2,\mathbf{up}}$, which (by hand calculation) is explicitly defined by conjunctions of degree 2 polynomial inequalities

$$\begin{aligned} A_{2,4} = \big\{(x_1, x_2) \in \mathbb{R}^2 \,|\, & x_1 < \delta \wedge x_2 > 1 + \delta \wedge \\ & x_1^2 + x_2^2 > (1+\delta)^2 \wedge \\ & -2b(1+2\delta) < 2bx_1 - x_2^2 \wedge \\ & 2bx_1 - x_2^2 < 2b\delta + \delta^2 - (1+2\delta)\big\}. \end{aligned}$$

Similarly, $A_{2,2}$ is $\delta\text{-}Unsafe_{2,\mathbf{dn}} \cap \delta\text{-}Unsafe_{2,\mathbf{ac}} \cap Safe_{2,\mathbf{up}}$, $A_{2,3}$ and $A_{2,5}$ are the two connected components of $Safe_{2,\mathbf{dn}} \cap Safe_{2,\mathbf{ac}} \cap Safe_{2,\mathbf{up}}$, and $A_{1,4}$ is $\delta\text{-}Unsafe_{1,\mathbf{dn}} \cap \delta\text{-}Unsafe_{1,\mathbf{ac}} \cap Safe_{1,\mathbf{up}}$. $A_{1,4}$ along with $A_{1,5}$ is too small to draw, so its location in Fig. 5 is indicated by $A_{1,*}$; likewise $A_{3,*}$ for $A_{3,4}$ and $A_{3,5}$. For each k, let $Danger_k$ be the all-δ-unsafe region $\cap_{c\in C} \delta\text{-}Unsafe_{k,c}$. In the example, $Danger_k = A_{k,6}$ is that part of A_k in the open annulus of inner and outer radii 1 and $1 + 2\delta$. Set $P = \{0\} \cup \{(k, m) | k \in K, m \in \{1, \cdots, 6\}\}$ to give a cover of 25 sets.

Stage 2—Determining the Control Modes and Controller I/O Relation: $Q \subseteq P \times C$. From the cover $\{A_p\}_{p\in P}$, the

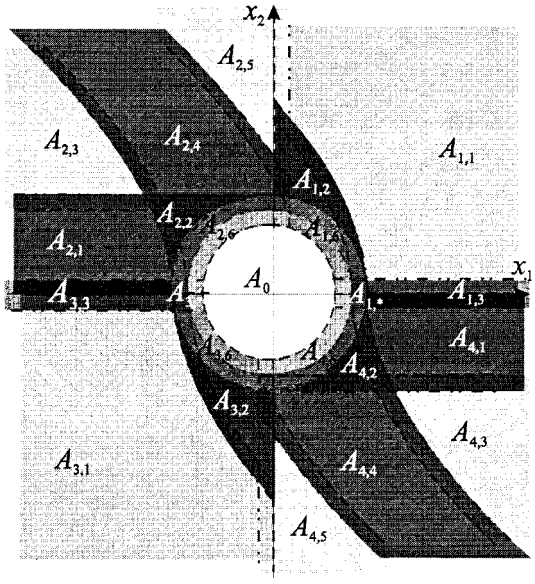


Fig. 5. Finite cover of $X = \mathbb{R}^2$.

first subtask is to determine a control law $Q_1 \subseteq P \times C$ adequate for the *safety* requirement. Our solution strengthens the safety condition a little more by using the set $Avoid := \cup_{k \in K} Danger_k \cup A_0$, which is the open disk of radius $1+2\delta$; we will ensure that the complement of $Avoid$ will be invariant. Take Q_1 to be the full product $P \times C$ minus any modes $((k,m),c)$ for which $\delta\text{-}Unsafe_{k,c} \subseteq A_{k,m}$. For example, $((2,2),\mathbf{dn})$, $((2,2),\mathbf{ac})$, and $((2,4),\mathbf{dn})$ are each discarded. For the danger regions $A_{k,6} \subseteq Avoid$, this will discard *all* modes $((k,6),c)$, so these need to be separately considered. In the example, we put back in to Q_1 the modes $((1,6),\mathbf{ac})$, $((2,6),\mathbf{up})$, $((3,6),\mathbf{ac})$, and $((4,6),\mathbf{dn})$, assigning default control actions in these danger zones. For the core failure region A_0, we extend the control alphabet C by adjoining a special symbol $\mathbf{fail}$, with the associated trivial flow $\phi_{\mathbf{fail}}(\mathbf{x}, t) = \mathbf{x}$ for all $t \in \mathbb{R}$ (i.e. $\dot{\mathbf{x}} = \mathbf{0}$), and put $(0, \mathbf{fail}) \in Q_1$.

The second subtask is to further refine Q_1 to a subset Q_2 which is adequate for the *event sequence* requirement. Control mode pairs $((k,m),c)$ must be discarded if c-evolution on $A_{k,m}$ leads directly to $A_{k'}$ where $k' \notin next(k)$ (for example, discard $((2,1),\mathbf{dn})$), or if c-evolution in $A_{k,m}$ never leaves $A_{k,m}$ and $k \notin next(k)$ (for example, discard $((1,1),\mathbf{ac})$ and $((1,1),\mathbf{up})$). For the positive content of this *until* property, Q_2 must be such that for each $((k,m_0),c_0) \in Q_2$, there is at least one sequence $((k,m_j),c_j)_{0 \le j \le N}$ in Q_2, with $m_j \neq m_i$ for $0 \le j < i \le N$, that defines a switching sequence (with no cycles in the sub-regions) that leads from A_{k,m_0} to $A_{k'}$ for some $k' \in next(k)$. The absence of such a sequence is the reason for discarding $((1,3),\mathbf{up})$ and $((2,3),\mathbf{dn})$. Before setting $Q := Q_2$, we must also check for coverage of the space X: for each region $p \in P$, we need at least one control action $c \in C$ such that $(p,c) \in Q_2$. This ensures that as an I/O relation, Q is *total*, and that in the final hybrid automaton H, for each $\mathbf{x} \in X$, there is a $q \in Q$ such that $(q,\mathbf{x}) \in \mathrm{dom}(h)$.

Stage 3—Determining the Control Graph: $E \subseteq Q \times Q$. We put $((p,c),(p',c')) \in E$ if $p' \neq p$, $A_p \cap A_{p'} \neq \emptyset$ and if $p' \neq 0$ and $p' \neq (k,m)$ where $A_{k,m} = Danger_k$, then c-evolution can lead from $A_p - A_{p'}$ into $A_p \cap A_{p'}$ (i.e., edges *into* regions within $Avoid$ only require overlap). For the *jump-infinite liveness* requirement, the graph E must be further checked to ensure that for each $(p,c) \in Q$, $p \neq 0$, every point in A_p can c-evolve into some overlap switching region $A_p \cap A_{p'}$, for some $((p,c),(p',c')) \in E$.

Our hybrid automaton solution H is illustrated in Fig. 6, with the labels for guards and resets omitted for readability. The required H-invariant set is $Good := X - Avoid$, and we further claim that all H-trajectories starting in $Good$ remain in A_k until they cross into $A_{k \ominus 1}$, and that all maximal H-trajectories starting in $Good$ are both jump-infinite and non-Zeno. Note that $Danger_k$ regions, with their default control actions, *could* cause pathologies, *if* a trajectory were ever to reach them. For example, an $\mathbf{ac}$ evolution from a point in $A_{1,6} \cap A_4$ would head straight for Bad. From points in $A_{2,2} \cap A_{2,6}$, we could produce a Zeno trajectory by switching between modes $((2,2),\mathbf{up})$ and $((2,6),\mathbf{up})$ after successive time durations $\Delta_i = 1/(N+i)^2$, for some sufficiently large N, so the trajectory would always remain in $A_{2,2} \cap A_{2,6}$.

We return to this example at various points in the rest of this paper. In particular, we will show how to succinctly characterize in modal logics the *operations on sets* used to construct the sets $\delta\text{-}Unsafe_{k,c}$ and $Safe_{k,c}$ from A_k and A_0, and to formalize the decisions required in the course of the construction, so those decisions could be resolved using a suitable model checking tool. A more detailed account of this synthesis procedure is given in a separate paper [32], and an extension that directly addresses robustness is given in [63].

III. Computational and Formal Logic Models

A. Labeled Transition Systems

We now turn to a more detailed examination of abstract transition system models, which provide both a formal *computational model* of system behavior and a *formal logic model* for the semantics of modal and temporal logics.

Definition 5: A *labeled transition system* (*LTS model* or *generalized Kripke model*) of signature (Σ, Φ) is a structure

$$\mathfrak{M} = \left(S, \{a^{\mathfrak{M}}\}_{a \in \Sigma}, \{[\![p]\!]^{\mathfrak{M}}\}_{p \in \Phi}\right)$$

where

- $S \neq \emptyset$ is the state space, of arbitrary cardinality;
- for each *relation label* (*transition* or *action label*) $a \in \Sigma$, $a^{\mathfrak{M}}$: $S \rightsquigarrow S$ is a relation on states;
- for each *atomic proposition* (*state predicate*, *event label*) $p \in \Phi$, $[\![p]\!]^{\mathfrak{M}} \subseteq S$ is a fixed subset of states.

The *signature* (Σ, Φ) of an LTS model $\mathfrak{M}$ is just the pair of alphabets, possibly infinite, indexing the relations and the state sets of $\mathfrak{M}$. Anticipating the logics in Section IV, where $a \in \Sigma$ and $p \in \Phi$ will occur in the formal syntax of the logic languages, the relation $a^{\mathfrak{M}}$ and the state set $[\![p]\!]^{\mathfrak{M}}$ are the semantic *denotations* in $\mathfrak{M}$ of the symbols a and p, respectively.

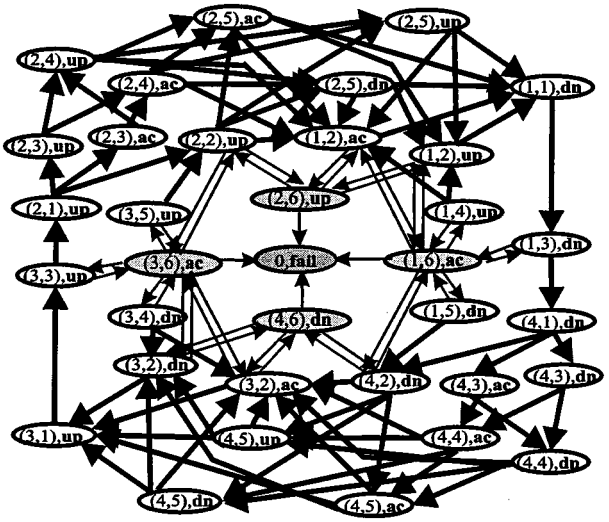


Fig. 6. Hybrid automaton solution for double integrator problem.

An LTS model is transparently a generalization of a finite automaton, obtained by merely dropping any assumptions of finiteness. The system transition relation δ: $S \times \Sigma \rightsquigarrow S$ for $\mathfrak{M}$ is the consolidation of the component transition relations: $s' \in \delta(s, a)$ iff $s \xrightarrow{a^{\mathfrak{M}}} s'$, and for the output map λ: $S \rightsquigarrow \Phi$, define $p \in \lambda(s)$ iff $s \in [\![p]\!]^{\mathfrak{M}}$; the output or *observation* at a state s is thus the set of all atomic propositions satisfied by s.

The discrete origins of transition system models are emphasized in the notation of [20], [21], [31], and [38], which use Q for the state space and write the global transition relation as a three-place relation $\rightarrow \subseteq Q \times \Sigma \times Q$. We modify the notation because we want to render transparent the generality of LTS models as abstract dynamical systems and the way in which trajectories and executions sequences are formed from the *composition* of the component transition relations.

Automata-theoretic approaches to formal methods are intimately related to logic-based work using linear temporal logics [64], [65] and overlap significantly with DES control theory [40], [66]. The common focus is on the *behavior* of an *abstract machine* as characterized by an automaton formal language of finite or infinite *words* over an alphabet, with words encoding the qualitative ordering of *events* and/or *actions*.

Definition 6: An *execution sequence* (computation sequence, path, run) of an LTS model $\mathfrak{M}$ is a finite or infinite sequence $\sigma = \langle s_i, a_i \rangle_{i \in I}$ over $S \times (\Sigma \cup \{\epsilon\})$ such that for all $i \in I$, either $s_i \xrightarrow{a_i^{\mathfrak{M}}} s_{1+1}$ or else $i = \max(I)$ and $a_i = \epsilon$, where $\epsilon \notin \Sigma$ is the empty word. (This is one way to treat finite and infinite sequences together.) The *observational trace* of σ is the sequence or word $\pi = \langle o_i, a_i \rangle_{i \in I}$ over the alphabet $\Gamma = \mathcal{P}(\Phi) \times (\Sigma \cup \{\epsilon\})$, where the ith observation $o_i = \lambda(s_i) = \{p \in \Phi | s_i \in [\![p]\!]^{\mathfrak{M}}\}$. For each state $s \in S$, the set $L(s) \subseteq \Gamma^* \cup \Gamma^\omega$ of traces of all execution sequences starting from s is called the *language* over Γ generated by s. When a set of initial states is given, via $\mathbf{Init} \in \Phi$, the language generated by $\mathfrak{M}$, written $L(\mathfrak{M})$, is the union of all languages $L(s)$ for $s \in [\![\mathbf{Init}]\!]^{\mathfrak{M}}$.

In automata-theoretic approaches to formal verification, properties of a machine $\mathfrak{M}$ are specified by another machine $\mathfrak{M}_{\text{spec}}$ of the same class (and signature). The verification question $\mathfrak{M} \models \varphi_{\text{spec}}$ in a logic-based approach is replaced by a question of *behavioral inclusion* between $\mathfrak{M}$ and $\mathfrak{M}_{\text{spec}}$; the basic relation is of *language inclusion*: $L(\mathfrak{M}) \subseteq L(\mathfrak{M}_{\text{spec}})$. For some automata-theoretic work, and for linear temporal logics such as LTL, the definition of $L(\mathfrak{M})$ is usually restricted to infinite words (ω-words) $\pi \in \Gamma^\omega$ [31], and the trace alphabet Γ may be just the observation/event alphabet $\Gamma = \mathcal{P}(\Phi)$, or the transition/action alphabet $\Gamma = \Sigma$.

B. LTS Models of Hybrid Systems

In representing a hybrid automaton H as an LTS model, one could, as a first pass, simply take the hybrid reachability relation h: $S_H \rightsquigarrow S_H$ as the sole transition relation over the state space S_H. With this relation, properties of the form "Some (all) H-trajectories starting from $s \ldots$" can be reexpressed in terms of relational successors, as "Some (all) h-successors of $s \ldots$," since a h-successor is any state lying on any H-trajectory starting from s. The temporal logic constructs "At some time…" and "At all times…" can be given a clean semantics over H-trajectories just using the h relation.

For both computational and conceptual reasons, this will not suffice. Computationally, the h relation is intractable since it will very rarely have an explicit first-order description; conceptually, it is a compound that needs analyzing. Going back to the definition of a hybrid trajectory, we need the notion of control action or flow applied within a prescribed region.

Definition 7: For any semiflow ϕ: $X \times \mathbb{R}^+ \rightarrow$ and set $A \subseteq X$, define a relation $e(A, \phi)$: $X \rightsquigarrow X$ of (positive) *evolution along ϕ within A* by

$$\mathbf{x} \xrightarrow{e(A,\phi)} \mathbf{x}' \stackrel{\text{def}}{\Leftrightarrow} (\exists t \in \mathbb{R}^+)\,[\mathbf{x}' = \phi(\mathbf{x}, t) \wedge (\forall s \in [0, t])\phi(\mathbf{x}, s) \in A]\,.$$

That is, $\mathbf{x}'$ is a $e(A, \phi)$-successor of $\mathbf{x}$ iff $\mathbf{x}'$ is a flow successor of $\mathbf{x}$ along ϕ and in addition, all intermediate points along the flow between $\mathbf{x}$ and $\mathbf{x}'$ lie inside A. A point $\mathbf{x}$ thus has either a continuum of successors under $e(A, \phi)$, namely, every point on the maximal integral curve of ϕ lying inside A and starting from $\mathbf{x}$, or it has just $\mathbf{x}$ itself, or it has none if $\mathbf{x} \notin A$. The (positive, or future) *orbit relation* $f(\phi)$: $X \rightsquigarrow X$ [58] is the unconstrained evolution relation: $f(\phi) \stackrel{\text{def}}{=} e(X, \phi)$. Direct from the flow laws, the relation $f(\phi)$ is *reflexive*, *transitive* and *weakly connected* (meaning that if $\mathbf{x} \xrightarrow{f(\phi)} \mathbf{x}_1$ and $\mathbf{x} \xrightarrow{f(\phi)} \mathbf{x}_2$ then either $\mathbf{x}_1 \xrightarrow{f(\phi)} \mathbf{x}_2$ or $\mathbf{x}_2 \xrightarrow{f(\phi)} \mathbf{x}_1$), and $e(A, \phi)$ shares the same properties except reflexivity is restricted to the domain A. The evolution relation can be further decomposed via the equation $e(A, \phi) = f(\phi) \cap (A \times A)$ exactly when the set A is *ϕ-convex*, in the sense that if $\mathbf{x}_1, \mathbf{x}_2 \in A$ and $\mathbf{x}_1 \xrightarrow{f(\phi)} \mathbf{x} \xrightarrow{f(\phi)} \mathbf{x}_2$, then $\mathbf{x} \in A$ [51]. For example, each

of the cover sets $A_{k,m} \subseteq X$ in the synthesis problem in Section II-D is convex with respect to each of the flows $\phi_{\mathbf{dn}}$, $\phi_{\mathbf{ac}}$, and $\phi_{\mathbf{up}}$.

Definition 8: An LTS model $\mathfrak{M}_H$ for a hybrid automaton H includes the following components:

- state space $S := S_H = Q \times X$;
- for each $q \in Q$, the *evolution* relation e_q: $S \rightsquigarrow S$

$$(l, \mathbf{x}) \xrightarrow{e_q} (l', \mathbf{x}') \overset{\text{def}}{\Leftrightarrow} l' = l = q \wedge \mathbf{x} \overset{e(Inv_q, \phi_q)}{\longrightarrow} \mathbf{x}'$$

- for each $(q, q') \in E$, the *reset* relation $r_{q,q'}$: $S \rightsquigarrow S$

$$(l, \mathbf{x}) \xrightarrow{r_{q,q'}} (l', \mathbf{x}') \overset{\text{def}}{\Leftrightarrow} l = q \wedge l' = q' \wedge \mathbf{x} \xrightarrow{R_{q,q'}} \mathbf{x}'$$

- a finite collection of distinguished subsets of S, including $[\![\mathbf{Inv}_q]\!]^{\mathfrak{M}_H} = \{q\} \times Inv_q$ for each $q \in Q$, and $[\![\mathbf{Grd}_{q,q'}]\!]^{\mathfrak{M}_H} = \{q\} \times Grd_{q,q'}$ for each $(q, q') \in E$, plus any other sets of states of interest for the particular system.

The transition alphabet Σ_H for $\mathfrak{M}_H$ includes symbols $\mathbf{e}_q$ for $q \in Q$ and $\mathbf{r}_{q,q'}$ for $(q, q') \in E$. It follows from the definition that $\text{dom}(e_q) = [\![\mathbf{Inv}_q]\!]^{\mathfrak{M}_H}$ and $\text{dom}(r_{q,q'}) = [\![\mathbf{Grd}_{q,q'}]\!]^{\mathfrak{M}_H}$.

The definition here is equivalent to that in [19], [21], and [31]. Close relatives of these LTS models include the *integration graphs* of [67]; the *generalized Kripke structures* of [26]; and the *phase transition systems* of [12]–[14]. The latter include a distinguished real-valued variable T for *global time*, with the coordinate dynamics $\dot{T} = 1$ within any evolution, and the identity constraint $T' = T$ in any reset relation.

We can now simply characterize the hybrid reachability relation.

Proposition 9: Given a hybrid automaton H over state space $S := S_H$, let e, r: $S \rightsquigarrow S$ be the finite unions $e = \cup_{q \in Q} e_q$ and $r = \cup_{(q,q') \in E} r_{q,q'}$ of the component evolution and reset relations of $\mathfrak{M}_H$. Then the H-reachability relation h: $S \rightsquigarrow S$ is such that

$$(q, \mathbf{x}) \xrightarrow{h} (q', \mathbf{x}') \quad \text{iff } (\exists k \in \mathbb{N})(q, \mathbf{x}) \xrightarrow{(er)^k e} (q', \mathbf{x}').$$

Hence, as a regular expression, $h = (er)^* e = e(re)^*$. ∎

The proof just appeals to the definition of the sequential composition (and union) of relations. For each H-trajectory $\eta = \langle \Delta_i, q_i, \gamma_i \rangle_{i \in I}$, there is a corresponding $\mathfrak{M}_H$ execution sequence $(q_0, \gamma_0(0)) \xrightarrow{e_{q_0}} (q_0, \gamma_0(\Delta_0)) \xrightarrow{r_{q_0,q_1}} (q_1, \gamma_1(0)) \xrightarrow{e_{q_1}} \cdots$ in alternating form. And conversely, for each $\mathfrak{M}_H$ execution sequence with alternating evolutions and resets, of the form $s_0 \xrightarrow{e_{q_0}} s_0' \xrightarrow{r_{q_0,q_1}} s_1 \xrightarrow{e_{q_1}} \cdots$, there is a unique H-trajectory which realizes this sequence.

The constrained evolution relations are called "time-abstract" (or better, "time-indeterminate"), since the time duration along the integral curve is "quantified out" [20]. This indeterminacy is exactly what is needed to capture the *event driven* nature of hybrid trajectories. Variations can be obtained by modifying the time quantification: *upper time-bounded* or *lower time-bounded* variants of $e(A, \phi)$ replace $(\exists t \in \mathbb{R}^+)$ by $(\exists t \in [0, \Delta])$ or $(\exists t \geq \Delta)$, respectively, for some constant $\Delta \in \mathbb{R}^+$, and an *exact-time* variant drops $(\exists t \in \mathbb{R}^+)$ and substitutes $t := \Delta$ in the body. The trajectories of systems with time delays between switching flows, as in [45], may be characterized using more complex regular expressions involving these time-bounded relations.

In our synthesis procedure in Section II-D, the initial data of the problem all live in the continuous world $X \subseteq \mathbb{R}^n$, and discrete states Q have to be constructed. This naturally leads to a "purely continuous" LTS model, called $\mathfrak{M}_C$, with $S := X$ the plant state space and atomic propositions naming the initial cover sets $\{A_k\}_{k \in K}$ and Bad. One can then work with evolution relations $e_{k,c} := e(A_k, \phi_c)$ and, once the final cover is constructed, reset relations $test.(A_{k,m} \cap A_{k',m'})$.

The example also illustrates how the very generic structure of an LTS model can be used to represent *static* or *structural* relations on a space, as well as dynamic transition relations. Recall our use of a *metric tolerance* parameter $\delta > 0$. Define a relation B_δ: $X \rightsquigarrow X$ by: $\mathbf{x} \xrightarrow{B_\delta} \mathbf{x}'$ iff $d(\mathbf{x}, \mathbf{x}') < \delta$. So the set image $B_\delta(\mathbf{x})$ is just the δ-ball around $\mathbf{x}$, and the relation B_δ is reflexive and symmetric, but not transitive. A point $\mathbf{x}$ lies in the set $\delta\text{-}Unsafe_{k,c}$ iff $\mathbf{x} \in A_k$ and there is an $\mathbf{x}' \in Bad$ that is a result of applying the composite relation $B_\delta \circ e_{k,c} \circ B_\delta$ to $\mathbf{x}$. We resume this discussion in Section IV-D.

The notion of continuity for automata developed by Arbib in [48, § 6.4], is based on an abstract *tolerance spaces* (X, ξ), for ξ any reflexive and transitive relation, rather than topological spaces. Our source for the notion of a tolerance relation, as was Arbib's, was work of the topologist Zeeman from the early 1960s.

C. Richer Formal Models of Hybrid Systems

Process algebra approaches to formal methods focus on the algebraic character of operations used in the formation of complex systems out of simpler ones, parallel composition being one such constructor. The work on HCSP in [28] extends Hoare's formalism of *communicating sequential processes* (CSPs) to include continuous evolution as a primitive process, in addition to discrete actions and asynchronous communication, and the process constructors include quantitative timing constructs. To make the connection with transition systems, the HCSP process expressions could be given a semantics as relations in an LTS model over a valuation space of continuous and discrete variables plus communication channels. The work in [29] uses Dijkstra's *predicate transformers* [68] to reason about the effect of actions or processes; these are essentially the same as the basic operators of modal logic, as discussed in Section IV below.

In work on discrete systems, there is a huge and well-established literature on the use of *petri nets* (in their many variations) for modeling systems consisting of a network of interacting subsystems in which the state is distributed; more recently, some of this work has been extended to timed and hybrid systems. The recent dissertation by Cook [69] is a substantial resource. In that work, a *hybrid net* model is given a formal representation as an LTS model, and property specification is given in the modal μ-calculus.

Other formal models of *open* or *reactive* hybrid systems, whose behavior is influenced by that of an external environment, are *hybrid I/O automata* (HIOA), introduced in [27] and used in this issue in [43], and *hybrid reactive modules* [70]. The state space for both these models is essentially of the form $S = X \times U \times V$, where X, U and V are the valuation spaces of *internal* or *private* variables, *input* or *control interface* variables, and *output* or *external* variables, respectively, and *actions* of such systems can represented as transition relations a: $S \rightsquigarrow S$ on the product state space.

IV. Property Specification Languages and Logics

A. Overview of Modal and Temporal Logics

The well-known temporal logics such as **LTL** or **CTL**, first formulated for program and hardware verification in the late 1970s and early 1980s in landmark papers [71], [72], belong to a larger and older family of *modal logics*. Modal logic was originally the province of philosophers interested in analyzing the concepts of *necessity* and *possibility*. Symbolic modal logics first appeared in 1912 in the work of Lewis, and modern approaches derive from the work of Kripke [73] in the early 1960s, who gave a formal semantics over models with a single "accessibility" relation between states referred to as "possible worlds;" these structures are known as *Kripke models*, and LTS models are their generalization to multiple relations. The survey articles [33], [37], [56], and [65], and the textbooks [30], [34], [35], and [74], are good resources for modal and temporal logics.

Fig. 7 is a schematic diagram of the family of logics with semantics over transition system models. The solid arrows indicate relations of inclusion or subsumption between logics, in the sense that everything expressible in the first can also be expressed in the second. Among logics with semantics over LTS models, indicated by boxes with solid outlines, the propositional μ-calculus $\mathbf{L}\mu$ [33], [36], [37] is the most expressive. Among the "nontemporal" modal logics, there is *propositional dynamic logic* **(PDL)** [75], [76], and modal logics for reasoning about the *knowledge* of an agent or process in a distributed system [74]. Boxes with broken outlines indicate logics that require some extension or adaption of LTS models. These are topological modal logics [23], real-time extensions of temporal logics [20], [21], [53], [77], interval temporal logics [15], [28], [78], [79], and *alternating temporal logic* **(ATL)** over game models [80]; the latter logic has an extension to an *alternating μ-calculus*, indicated by the broken line arrow.

Since virtually all the work on logic-based specification of hybrid systems, and discrete systems before them, has been within the narrower subfamily of temporal logics, our break with tradition needs some further explanation.

In Pnueli's landmark paper [71], he identified the then new temporal logics as falling under an *endogenous* or "internal" approach to property specification. In branching temporal logics such as **CTL** or **CTL*** (reviewed in this issue in [31], *next-step* formulas $\exists \bigcirc \varphi$ have the semantics: "*Some*

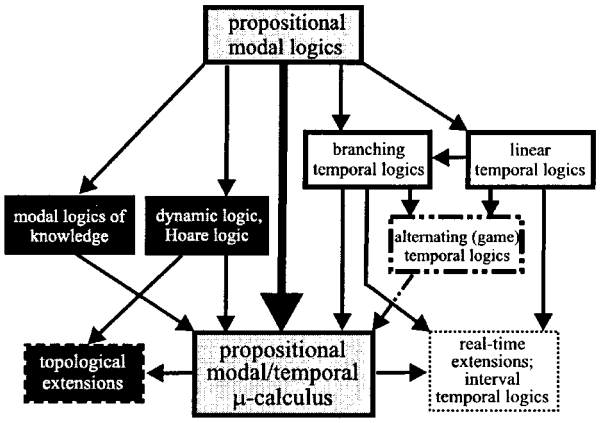


Fig. 7. Family of propositional modal and temporal logics.

one-step successor satisfies φ," where the one-step relation is $\xrightarrow{\mathfrak{M}} = \cup_{a \in \Sigma} a^{\mathfrak{M}}$ and the component relations are abstracted away. Behavior along execution sequences is captured by taking the Kleene star of the one-step relation, and the *always* formula $\forall \Box \varphi$ has the semantics: "Along *all* execution sequences, *all* states satisfy φ." The key point is that in temporal logics, one can reason directly about execution sequences of a single system, but there is no facility within their formal languages to talk about how the behavior of any one system is *composed* from its internal parts, or to *compare* the behavior of two or more systems.

The endogenous approach is in contrast with the *exogenous* or "external" approach, exemplified by the polymodal logic **PDL** [75], [76], in which the component transition relations of LTS models are "first-class objects," explicitly named in the syntax of the logic. In **PDL**, one reasons directly about the primitive relations of a model, and compound relations formed from them using the regular expression constructors of composition, finite union and Kleene star, and others such as the $test$ constructor. While the exogenous approach has been highly successful for purely discrete systems, where the component transition relations have a homogeneous character, it is worthy of reexamination in the case of *hybrid* systems, where the internal components are necessarily *heterogenous* in nature.

We start with an exposition of the base logic, *propositional poly-modal logic* (**PML**), which can be taken as the common core of all modal and temporal logics over LTS models, and illustrate how to use it to express finitary properties of relations, and in particular, various steps in our synthesis procedure from Section II-D. We then turn to $\mathbf{L}\mu$, which adds to **PML** the power to reason about infinitary constructions of relations such as the Kleene star (so subsuming **PDL**), as well as all the operators of temporal logics, and illustrate how it can be used to cleanly and simply express a wide variety of properties of hybrid automata. We also survey the literature on temporal and modal logics for hybrid and timed systems, with a focus on quantitative real-time properties beyond what is expressible in $\mathbf{L}\mu$, and on topological extensions of **PML** and $\mathbf{L}\mu$.

B. Propositional Poly-Modal Logic

Definition 10: The set $\mathcal{S}(\Sigma, \Phi)$ of formulas of **PML** in the signature (Σ, Φ) is inductively defined by the grammar

$$\varphi ::= p \,|\, \neg\varphi \,|\, \varphi_1 \vee \varphi_2 \,|\, \langle a\rangle\varphi$$

for atomic propositions $p \in \Phi$ and relation labels $a \in \Sigma$.

The other propositional connectives can be defined in a standard way: conjunction $\varphi_1 \wedge \varphi_2 \stackrel{\text{def}}{=} \neg(\neg\varphi_1 \vee \neg\varphi_2)$, implication $\varphi_1 \rightarrow \varphi_2 \stackrel{\text{def}}{=} (\neg\varphi_1 \vee \varphi_2)$, and equivalence $\varphi_1 \leftrightarrow \varphi_2 \stackrel{\text{def}}{=} (\varphi_1 \rightarrow \varphi_2) \wedge (\varphi_2 \rightarrow \varphi_1)$. In addition, we introduce $\mathbf{tt}$ ("*true*") as an abbreviation for any fixed propositional tautology, such as $p \vee \neg p$ for some $p \in \Phi$, and "*false*" $\mathbf{ff} \stackrel{\text{def}}{=} \neg\mathbf{tt}$. The modal operators $\langle a\rangle$ are pronounced "*diamond-a*," and the dual "*box-a*" operators are defined by $[a]\varphi \stackrel{\text{def}}{=} \neg\langle a\rangle\neg\varphi$. In the syntax of the branching temporal logic **CTL** [31], the single *next-step* operator $\exists\bigcirc$ replaces all the separate modal operators $\langle a\rangle$ for $a \in \Sigma$, and a single $\forall\bigcirc$ replaces all the operators $[a]$.

Semantically, a formula φ will denote a set $[\![\varphi]\!]^{\mathfrak{M}} \subseteq S$ of states. The Boolean operations of negation and disjunction clearly correspond to the set-theoretic operations of complement and union. For modal formulas $\langle a\rangle\varphi$ and $[a]\varphi$, we need operators built from the relations $a^{\mathfrak{M}}\colon S \rightsquigarrow S$.

Analogous to the inverse-image operator of a single-valued function, any relation $r\colon X \rightsquigarrow Y$ determines a dual pair of *preimage operators* $\mathrm{Pre}^{\exists}(r)$, $\mathrm{Pre}^{\forall}(r)$: $\mathcal{P}(Y) \rightarrow \mathcal{P}(X)$ mapping sets to sets, defined by

$$\mathrm{Pre}^{\exists}(r)(A) \stackrel{\text{def}}{=} \left\{x \in X \,|\, (\exists y \in Y)\left[x \xrightarrow{r} y \wedge y \in A\right]\right\}$$
$$\mathrm{Pre}^{\forall}(r)(A) \stackrel{\text{def}}{=} \left\{x \in X \,|\, (\forall y \in Y)\left[x \xrightarrow{r} y \Rightarrow y \in A\right]\right\}.$$

In words, $x \in \mathrm{Pre}^{\exists}(r)(A)$ iff there is *some* r-successor of x that lies in A, while $x \in \mathrm{Pre}^{\forall}(r)(A)$ iff *all* r-successors of x lie in A. Note that the latter still holds when $x \notin \mathrm{dom}(r)$, so there are *no* r-successors of x, hence $X - \mathrm{dom}(r) \subseteq \mathrm{Pre}^{\forall}(r)(A)$ for any set A. The duality between the operators is with respect to set complement, and is given by: $\mathrm{Pre}^{\forall}(r)(A) = X - \mathrm{Pre}^{\exists}(r)(Y - A)$.

The preimage operators have appeared under various names and notations, and in diverse settings, throughout mathematics and computer science. From Dijkstra's famous text [68], they are known as *predicate transformers*. In that work, the set $\mathrm{Pre}^{\forall}(r)(A)$ is referred to as $\mathrm{wlp}_r(A)$, the *weakest liberal precondition* of A under a relation $r\colon X \rightsquigarrow X$, while the *weakest precondition* is taken as $\mathrm{wp}_r(A) \stackrel{\text{def}}{=} \mathrm{Pre}^{\forall}(r)(A) \cap \mathrm{dom}(r)$, where $\mathrm{dom}(r) = \mathrm{Pre}^{\exists}(r)(X)$. A related transformer is the *postimage operator* $\mathrm{Post}^{\exists}(r) \stackrel{\text{def}}{=} \mathrm{Pre}^{\exists}(\breve{r})$, also called the *direct-image* $r(A)$. For Dijkstra, this is $\mathrm{sp}_r(A)$, the *strongest postcondition* of A under r. In general topology, the purely topological notions of *continuity* for relations/set-valued maps $r\colon X \rightsquigarrow Y$ are defined using the operators $\mathrm{Pre}^{\exists}(r)$ and $\mathrm{Pre}^{\forall}(r)$ [57], [60]; we return to point this in Section IV-H.

Definition 11: For formulas $\varphi \in \mathcal{S}(\Sigma, \Phi)$, the *denotation set* $[\![\varphi]\!]^{\mathfrak{M}} \subseteq S$ in an LTS model $\mathfrak{M}$ of signature (Σ, Φ) is defined by induction on the structure of formulas. For the base case of $p \in \Phi$, the denotation sets $[\![p]\!]^{\mathfrak{M}}$ are given as components of $\mathfrak{M}$, and for compound formulas

$$[\![\neg\varphi]\!]^{\mathfrak{M}} \stackrel{\text{def}}{=} S - [\![\varphi]\!]^{\mathfrak{M}}$$
$$[\![\varphi_1 \vee \varphi_2]\!]^{\mathfrak{M}} \stackrel{\text{def}}{=} [\![\varphi_1]\!]^{\mathfrak{M}} \cup [\![\varphi_2]\!]^{\mathfrak{M}}$$
$$[\![\langle a\rangle\varphi]\!]^{\mathfrak{M}} \stackrel{\text{def}}{=} \mathrm{Pre}^{\exists}\left(a^{\mathfrak{M}}\right)\left([\![\varphi]\!]^{\mathfrak{M}}\right) \quad \text{for } a \in \Sigma.$$

For formulas $\varphi \in \mathcal{S}(\Sigma, \Phi)$, we say: φ is *satisfied* at state s in $\mathfrak{M}$, written $\mathfrak{M}, s \models \varphi$, if $s \in [\![\varphi]\!]^{\mathfrak{M}}$, and φ is *true* in $\mathfrak{M}$, written $\mathfrak{M} \models \varphi$, if $[\![\varphi]\!]^{\mathfrak{M}} = S$.

Let $\mathcal{B}(\mathfrak{M}) \stackrel{\text{def}}{=} \{[\![\varphi]\!]^{\mathfrak{M}} \in \mathcal{P}(S) | \varphi \in \mathcal{S}(\Sigma, \Phi)\}$ be the family of all sets of states in $\mathfrak{M}$ denoted by PML formulas. Then $\mathcal{B}(\mathfrak{M})$ forms a *Boolean algebra* of sets, which is generated from the atomic state sets $[\![p]\!]^{\mathfrak{M}}$ for $p \in \Phi$. The "top" element of the Boolean algebra is $[\![\mathbf{tt}]\!]^{\mathfrak{M}} = \mathbf{S}$, the "bottom" element is $[\![\mathbf{ff}]\!]^{\mathfrak{M}} = \varnothing$, and it is partially ordered by inclusion $\subseteq$, which corresponds to the implication connective in the sense that $\mathfrak{M} \models (\varphi \rightarrow \psi)$ iff $[\![\varphi]\!]^{\mathfrak{M}} \subseteq [\![\psi]\!]^{\mathfrak{M}}$.

The algebraic theory of relations and their operators on sets was developed in the 1940s and 1950s in the work of Tarski and Jónsson [81], [82]. In terms of that work, the algebra $\mathcal{B}(\mathfrak{M})$ is a *Boolean algebra with operators*, which is closed under $\mathrm{Pre}^{\exists}(a^{\mathfrak{M}})$ [and hence also $\mathrm{Pre}^{\forall}(a^{\mathfrak{M}})$] for each $a \in \Sigma$. In more modern terms, $\mathcal{B}(\mathfrak{M})$ is a *modal algebra* [33], [51], the smallest of all modal algebras of sets $\mathcal{B} \subseteq \mathcal{P}(S)$ for $\mathfrak{M}$, and we refer to it as the *minimal modal algebra* for $\mathfrak{M}$.

C. Finitary Relational Properties in PML

The axioms for a Hilbert-style proof system for **PML** consist of the axioms of (classical) propositional logic **PL** (see [34]) plus the following:

$$Ax.\langle a\rangle\vee\colon\ \langle a\rangle(\varphi_1 \vee \varphi_2) \leftrightarrow (\langle a\rangle\varphi_1 \vee \langle a\rangle\varphi_2)$$
$$Ax.\langle a\rangle\mathbf{ff}\colon\ \langle a\rangle\mathbf{ff} \leftrightarrow \mathbf{ff}.$$

The corresponding axioms $Ax.[a]\wedge$ and $Ax.[a]\mathbf{tt}$ are obtained by Boolean duality. The inference rules for **PML** consist of the classical *modus ponens*, $Rule.$MP: from $\psi \rightarrow \varphi$ and ψ, infer φ, and the rule of *modal necessitation*, $Rule.$Nec: from φ, infer $[a]\varphi$. (The *soundness* of the latter rule says if $[\![\varphi]\!]^{\mathfrak{M}} = S$ then $[\![[a]\varphi]\!]^{\mathfrak{M}} = S$.) These axioms and rules characterize *normal* modal operators [35]. A dominant theme within general modal logic is the study of the correspondence between elementary properties of binary relations and formulas of modal logic [35], [56]. For example, the properties of *reflexivity*, *transitivity*, and *weak connectedness*, as possessed by the orbit relation $f(\phi)$ of a flow, are characterized by the formula schemes $\mathbf{T}$, $\mathbf{4}$, and $\mathbf{L}$ (the names being historical within modal logic)

$$\langle a\rangle\mathbf{T}\colon\quad \varphi \rightarrow \langle a\rangle\varphi$$
$$\langle a\rangle\mathbf{4}\colon\quad \langle a\rangle\langle a\rangle\varphi \rightarrow \langle a\rangle\varphi$$
$$\langle a\rangle\mathbf{L}\colon\quad \langle a\rangle\varphi_1 \wedge \langle a\rangle\varphi_2 \rightarrow (\langle a\rangle(\varphi_1 \wedge \varphi_2) \vee \langle a\rangle(\varphi_1 \wedge \langle a\rangle\varphi_2) \vee \langle a\rangle(\varphi_2 \wedge \langle a\rangle\varphi_1)).$$

Together with the normality axioms, these three formula schemes axiomatize the modal logic **S4.3**.

The preimage operators are well behaved with respect to the regular expression constructors of *sequential composition* and *finite unions*, justifying the definitional extension of **PML** by $\langle ab\rangle\varphi \stackrel{\text{def}}{=} \langle a\rangle\langle b\rangle\varphi$ and $\langle a \cup b\rangle\varphi \stackrel{\text{def}}{=} \langle a\rangle\varphi \vee \langle b\rangle\varphi$. We can also conservatively extend **PML** with defined modalities for compound relation symbols $(\psi?)$ formed from formulas ψ, with the semantics $(\psi?)^{\mathfrak{M}} = test.[\![\psi]\!]^{\mathfrak{M}}$. In the syntax of **PML**, define $\langle\psi?\rangle\varphi \stackrel{\text{def}}{=} \psi \wedge \varphi$.

The expressiveness of **PML** can be properly increased by adjoining modalities $\langle \breve{a}\rangle$ and $[\breve{a}]$ for the converse relations, interpreted by the postimage operators; the temporal logic analogs are referred to as *past* operators. The additional axiom schemes are

$$Ax.\langle\breve{a}\rangle - 1: \quad \varphi \to [a]\langle\breve{a}\rangle\varphi$$
$$Ax.\langle\breve{a}\rangle - 2: \quad \langle\breve{a}\rangle[a]\varphi \to \varphi.$$

In work on deductive verification using temporal logic, such as [12]–[14], and [30], it is standard practice to supplement the syntax of temporal logic with additional notation to express *safety verification* or *correctness conditions*. The so-called *Hoare triple* notation $\{\psi\}a\{\varphi\}$ is transcribed into the language of **PML** by the formula $\psi \to [a]\varphi$, which reads: "If ψ holds, then all a-successors satisfy φ;" in particular, $\varphi \to [a]\varphi$ asserts that the set of φ-states is (future-)*invariant* under the relation $a^{\mathfrak{M}}$.

Working in an LTS model $\mathfrak{M} = \mathfrak{M}_H$ of a hybrid automaton H, we have $\langle \mathbf{e}_q\rangle \mathbf{tt} \leftrightarrow \mathbf{Inv_q}$ and $\langle \mathbf{r}_{\mathbf{q},\mathbf{q}'}\rangle \mathbf{tt} \leftrightarrow \mathbf{Grd}_{\mathbf{q},\mathbf{q}'}$. The formula $\mathbf{Grd}_{q,q'} \to [\mathbf{r}_{q,q'}]\mathbf{Inv}_{q'}$ asserts that H meets the sensible design condition that resets under $r_{q,q'}$ always lead to $Inv_{q'}$. Extending the signature of $\mathfrak{M}_H$ by adjoining relations f_q for the *unconstrained* flow or orbit relations on $S = Q \times X$ (replacing $e(Inv_q, \phi_q)$ with $f(\phi_q)$ in the definition of e_q in Definition 8), the formula $\mathbf{Inv}_q \to \langle \mathbf{f}_q\rangle\neg\mathbf{Inv}_q$ asserts that the flow ϕ_q cannot stay in Inv_q forever.

Within **PML**, we are limited to the expression of properties of H-trajectories with a *finite* number of jumps. Let $\mathbf{e}$ and $\mathbf{r}$ denote the relational unions e and r, respectively, of the component evolution and reset relations. Then for a fixed $k \in \mathbb{N}$, the modal formula $\langle(\mathbf{er})^k\mathbf{e}\rangle\varphi$ denotes the set of states $(q, x) \in S$ from which there is *some* H-trajectory with k discrete jumps that reaches a φ-state, while $[(\mathbf{er})^k\mathbf{e}]\varphi$ denotes the set of states $(q, x) \in S$ from which *every* H-trajectory with k discrete jumps reaches the set of φ-states, and remains there throughout its final evolution interval. In order to reason about *arbitrary* H-trajectories, and to define modalities $\langle\mathbf{h}\rangle$ and $[\mathbf{h}]$, which correspond to the preimage operators of the H-reachability relation $h = (er)^*e$, we have to move to the μ-calculus.

D. Example Control Problem Revisited

Before making the infinitary move, we return to the example synthesis problem in Section II-D. As discussed at the end of Section III-B, we can work in an LTS model $\mathfrak{M}_C$ with $S := X = \mathbb{R}^2$. Define the semantics of relation symbols by $(\mathbf{e}_{k,c})^{\mathfrak{M}_C} := e(A_k, \phi_c)$ and $(\delta)^{\mathfrak{M}_C} := B_\delta$. Then the sets $\delta\text{-}Unsafe_{k,c}$ and $Safe_{k,c}$ have the modal characterizations

$$\delta\text{-}\mathbf{Unsafe}_{k,c} \stackrel{\text{def}}{=} \mathbf{A}_k \wedge \langle\delta\rangle\langle\mathbf{e}_{k,c}\rangle\langle\delta\rangle\mathbf{Bad}$$
$$\mathbf{Safe}_{k,c} \stackrel{\text{def}}{=} \mathbf{A}_k \wedge [\mathbf{e}_{k,c}]\neg\langle\delta\rangle\mathbf{Bad}.$$

This is because $\langle\delta\rangle\varphi$ denotes the set of points within δ of some φ-state, so the δ-"closure," while the dual $[\delta]\varphi$ denotes the δ-"interior," meaning the set of points all of whose δ-neighbors are still φ-states. The "fattening" with the outer $\langle\delta\rangle$ operator in $\delta\text{-}\mathbf{Unsafe}_{k,c}$ creates the overlaps. The subregions $A_{k,m}$ are then modally defined by conjunctions of $\delta\text{-}\mathbf{Unsafe}_{k,c}$'s and $\mathbf{Safe}_{k,c}$'s. The nominated invariant set $Good$ is characterized by

$$\mathbf{Good} \stackrel{\text{def}}{=} \neg\langle\delta\rangle\langle\delta\rangle\mathbf{Bad}$$

The fact that each of the cover sets $A_{k,m} \subseteq X$ are convex with respect to each of the flows ϕ_c is expressed by the formula

$$\mathbf{A}_{k,m} \to [\mathbf{f}_c](\langle\mathbf{f}_c\rangle\mathbf{A}_{k,m} \to \mathbf{A}_{k,m})$$

where $(\mathbf{f}_c)^{\mathfrak{M}} := f(\phi_c)$, and likewise for the initial quadrant regions A_k. This in turn implies

$$\langle\mathbf{e}_{(k,m),c}\rangle\varphi \leftrightarrow \mathbf{A}_{k,m} \wedge \langle\mathbf{f}_c\rangle(\varphi \wedge \mathbf{A}_{k,m})$$
$$[\mathbf{e}_{(k,m),c}]\,\varphi \leftrightarrow \mathbf{A}_{k,m} \to [\mathbf{f}_c](\mathbf{A}_{k,m} \to \varphi).$$

The safety control law $Q_1 \subseteq P \times C$ is constructed so as to ensure that for each $((k,m),c) \in Q_1$

$$\mathbf{Good} \to [\mathbf{e}_{(k,m),c}]\,\mathbf{Good}$$

is true in $\mathfrak{M}_C$. In refining Q_1 to deal with the event sequence requirement, the reason for discarding $((1,1),\mathbf{up})$ is because $\mathbf{A}_{1,1} \to [\mathbf{f_{up}}]\mathbf{A}_{1,1}$ is true in $\mathfrak{M}_C$, which means $A_{1,1}$ is future-invariant under the flow $\phi_{\mathbf{up}}$. The mode $((2,1),\mathbf{dn})$ is discarded because the set $[\![\mathbf{A}_{2,1} \wedge \langle\mathbf{e}_{(2,1),\mathbf{dn}}\rangle\mathbf{A}_3]\!]^{\mathfrak{M}_C}$ is nonempty. For the positive content of the event sequence requirement, we have to produce a subset $Q_2 \subseteq Q_1$ such that for each $((k,m_0),c_0) \in Q_2$, there is at least one switching sequence $\langle(k, m_j), c_j\rangle_{0\le j\le N}$ starting from A_{k,m_0} in control c_0 and leading through A_k and into $A_{k\ominus 1}$, with no cycles in the subregions. For example, for each $c \in \{\mathbf{dn}, \mathbf{ac}, \mathbf{up}\}$, we keep $((2,5),c)$ because

$$\mathbf{A}_{2,5} \to [\mathbf{e}_{(2,5),c}]\,\langle\mathbf{e}_{(2,5),c}\rangle\mathbf{A}_1$$

is true in $\mathfrak{M}_C$; this says c-evolution in $A_{2,5}$ *inevitably* leads to A_1. Then we can keep $((2,4),\mathbf{ac})$ because

$$\mathbf{A}_{2,4} \to [\mathbf{e}_{(2,4),\,\mathbf{ac}}]\,\langle\mathbf{e}_{(2,4),\mathbf{ac}}\rangle\,(\mathbf{A}_1 \vee \mathbf{A}_{2,5}).$$

Each quadrant A_k can be systematically searched, starting with the subregions $A_{k,m}$ that overlap with $A_{k\ominus 1}$.

E. Propositional Modal μ-Calculus $\mathbf{L}\mu$

The *propositional modal μ-calculus* $\mathbf{L}\mu$ [36] is a logic extending **PML** by adjoining least (μ) and greatest (ν) *fixed-point quantifiers*. Semantically, this adds the mathematically

common and highly useful construct "the *least* set A such that $Op(A) = A$" or "the *greatest* set A such that $Op(A) = A$," where Op: $\mathcal{P}(S) \to \mathcal{P}(S)$ is an inclusion-monotone function mapping sets to sets. This fixed-point construct is the essence of the notion of *inductive definability*, including the iteration construct of the Kleene-star a^* of regular expressions. The μ-calculus occupies a paramount place in formal methods: the expressive power of the fixed-point quantifiers are such that it subsumes virtually all modal and temporal logics over LTS models, and formal verification by *model checking* for all such logics is essentially based on a translation into the μ-calculus [37]. In DES control theory, the *modular feedback logic* in [39] consists of propositional logic, Dijkstra's predicate transformers, and least and greatest fixed points.

Definition 12: Let PVar be a set of *propositional variables* (*second-order* or *set-valued* variables). The set $\mathcal{F}_\mu(\Sigma, \Phi)$ of formulas of $\mathbf{L}\mu$ in the signature (Σ, Φ) is inductively defined by the grammar

$$\varphi ::= p \,|\, Z \,|\, \neg\varphi \,|\, \varphi_1 \vee \varphi_2 \,|\, \langle a\rangle\varphi \,|\, \mu Z.\varphi$$

for atomic propositions $p \in \Phi$, propositional variables $Z \in \mathrm{PVar}$, and relation labels $a \in \Sigma$, with a further syntactic restriction that $\mu Z.\varphi$ is in $\mathcal{F}_\mu(\Sigma, \Phi)$ only if every *free* occurrence of Z in φ is within the scope of an even number of negations. Analogous with first-order quantification, a variable Z in the scope of μZ or νZ is said to be *bound*, and *free* otherwise. A formula $\varphi \in \mathcal{F}_\mu(\Sigma, \Phi)$ will be called a *sentence* if it contains no free variables, and the set of all sentences will be denoted $\mathcal{S}_\mu(\Sigma, \Phi)$.

Propositional variables $Z \in \mathrm{PVar}$ are our means to talk about *any* or *all* subsets of states $A \in \mathcal{P}(S)$, in addition to particular constant sets $[\![p]\!]^{\mathfrak{M}} \subseteq S$. The fixed-point constructors give a special kind of *quantification* over subsets of states. Formulas $\mu Z.\varphi$ are informally read "The *smallest* set Z such that $Z = \varphi(Z)$," while the dual $\nu Z.\varphi \stackrel{\text{def}}{=} \neg\mu Z.\neg\varphi[Z := \neg Z]$ is read "The *largest* set Z such that $Z = \varphi(Z)$." The expression $\varphi[Z := \psi]$ means the formula resulting from φ by substituting ψ for all free occurrences of Z, with a side condition to avoid unintended clashes of variable names.

Definition 13: A *variable assignment* in an LTS model $\mathfrak{M} = (S, \{a^{\mathfrak{M}}\}_{a\in\Sigma}, \{[\![p]\!]^{\mathfrak{M}}\}_{p\in\Phi})$, is any function ξ: PVar $\to \mathcal{P}(S)$. For formulas $\varphi \in \mathcal{F}_\mu(\Sigma, \Phi)$ and variable assignments ξ, the *denotation set* $[\![\varphi]\!]^{\mathfrak{M}}_\xi \subseteq S$ in an LTS model $\mathfrak{M}$ of signature (Σ, Φ) is defined by induction on the structure of formulas. For the propositional connectives and modal operators in $\mathbf{L}\mu$, the semantic clauses are as for **PML**, with the addition of a subscript ξ

$$\begin{aligned}
[\![p]\!]^{\mathfrak{M}}_\xi &\stackrel{\text{def}}{=} [\![p]\!]^{\mathfrak{M}} \\
[\![Z]\!]^{\mathfrak{M}}_\xi &\stackrel{\text{def}}{=} \xi(Z) \\
[\![\mu Z.\varphi]\!]^{\mathfrak{M}}_\xi &\stackrel{\text{def}}{=} \bigcap \left\{ A \in \mathcal{P}(S) \,|\, [\![\varphi]\!]^{\mathfrak{M}}_{\xi(A/Z)} \subseteq A \right\}
\end{aligned}$$

where $\xi(A/Z)$ is the assignment that is the same as ξ except for assigning the set A to the variable Z. For formulas $\varphi \in \mathcal{F}_\mu(\Sigma, \Phi)$ and assignments ξ: PVar $\to \mathcal{P}(S)$ in $\mathfrak{M}$, we say: φ is *satisfied* at state s in $(\mathfrak{M}, \xi)$, written $\mathfrak{M}, \xi, s \models \varphi$, if $s \in [\![\varphi]\!]^{\mathfrak{M}}_\xi$, and φ is *true* in $\mathfrak{M}$, written $\mathfrak{M} \models \varphi$, if $[\![\varphi]\!]^{\mathfrak{M}}_\xi = S$ for *all* assignments ξ in $\mathfrak{M}$.

For sentences $\varphi \in \mathcal{S}_\mu(\Sigma, \Phi)$, their denotation is independent of any variable assignment, and so written $[\![\varphi]\!]^{\mathfrak{M}}$. Model checking for $\mathbf{L}\mu$ applies only to sentences since the denotation $[\![\varphi]\!]^{\mathfrak{M}}$ has to be computed. Note that **PML** formulas are all $\mathbf{L}\mu$ sentences; i.e., $\mathcal{S}(\Sigma, \Phi) \subseteq \mathcal{S}_\mu(\Sigma, \Phi)$.

The formal semantics say the set $[\![\mu Z.\varphi]\!]^{\mathfrak{M}}_\xi$ is the *least prefixed point* (the *intersection* of all such prefixed points) of the operator on sets $\varphi(Z)^{\mathfrak{M}}_\xi$: $\mathcal{P}(S) \to \mathcal{P}(S)$ given by $(\varphi(Z)^{\mathfrak{M}}_\xi)(A) \stackrel{\text{def}}{=} [\![\varphi]\!]^{\mathfrak{M}}_{\xi(A/Z)}$. The syntactic restriction on μ-formulas ensures that this operator is $\subseteq$-monotone. The Tarski–Knaster theorem for monotone operators on complete lattices [such as $\mathcal{P}(S)$] guarantees that the least pre-fixed point *exists*, and is equal to the least fixed point.

In order to try to *compute* the denotation of a fixed-point formula, as is required for model checking, one appeals to the Hitchcock–Park fixed-point theorem. This result says the set $[\![\mu Z.\varphi]\!]^{\mathfrak{M}}_\xi$ may be characterized as a *union* of an $\subseteq$-increasing chain of *approximations*, starting with $\emptyset$, and formed by iterating the $\varphi(Z)^{\mathfrak{M}}_\xi$ operator until a fixed-point set is reached. The finite stages of the approximation sequence are explicitly described by the denotations of formulas $[\![\varphi^k]\!]^{\mathfrak{M}}_\xi$, where the "unwinding" sequence is recursively defined by

$$\varphi^0 \stackrel{\text{def}}{=} \mathbf{ff} \quad \text{and} \quad \varphi^{k+1} \stackrel{\text{def}}{=} \varphi[Z := \varphi^k] \text{ for } k \in \mathbb{N}.$$

The approximation up to stage ω (the *ordinal number* of $\mathbb{N}$) is the union over $k \in \mathbb{N}$, but in general, the approximation sequence may proceed past ω and through *transfinite ordinals*, of cardinality less than or equal to that of S, before convergence occurs. There is a well-developed theory of approximations of fixed points; for our purposes, it suffices to know that when the semantic operator $\varphi(Z)^{\mathfrak{M}}_\xi$ distributes over unions of countable $\subseteq$-increasing chains of sets (a property also called "$\cup$-*continuous*" [33], [52]), the ordinal of convergence for $\mu Z.\varphi$ is at worst ω. In particular, for the Kleene star constructor on relations, which is definable in the μ-calculus by $\langle a^*\rangle\varphi \stackrel{\text{def}}{=} \mu Z.\varphi \vee \langle a\rangle Z$, one has

$$[\![\langle a^*\rangle\varphi]\!]^{\mathfrak{M}}_\xi = \bigcup_{k\in\mathbb{N}} [\![\langle a^k\rangle\varphi]\!]^{\mathfrak{M}}_\xi.$$

The notion of a *bisimulation relation* on or between LTS models is of fundamental importance, and a central concern of [31] in this special issue. A bisimulation equivalence is a type of *congruence* that respects the component relations and distinguished subsets of an LTS model. The fundamental property of truth preservation is the following.

Proposition 14 ([33]): Given an LTS model $\mathfrak{M}$ of signature (Σ, Φ), if $\approx$ is a bisimulation equivalence on $\mathfrak{M}$, then for all sentences $\varphi \in \mathcal{S}_\mu(\Sigma, \Phi)$ and all states $s, s' \in S$

$$s \approx s' \;\Rightarrow\; \left[s \in [\![\varphi]\!]^{\mathfrak{M}} \Leftrightarrow s' \in [\![\varphi]\!]^{\mathfrak{M}} \right]. \qquad \blacksquare$$

A corollary is that if $\mathfrak{M}$ has a bisimulation equivalence of *finite* index N, then for each sentence $\mu Z.\varphi$, its approx-

imation sequence is guaranteed to converge at some stage $k \leq N$, so

$$[\![\mu Z.\varphi]\!]^{\mathfrak{M}} = [\![\varphi^k]\!]^{\mathfrak{M}} = [\![\varphi^{k+1}]\!]^{\mathfrak{M}}.$$

This is because Proposition 14 entails that the denotation set of each $\mathbf{L}\mu$ sentence must be a union of equivalence classes. The quotient LTS model $\mathfrak{M}_{\approx}$ is then a fully discrete finite automaton *simulacrum* of the original $\mathfrak{M}$, which satisfies and makes true all the same μ-calculus sentences.

F. Formalizing Properties of Hybrid Automata in $\mathbf{L}\mu$

The modal μ-calculus provides a very rich formalism in which to formally express properties of hybrid automata. With the use of defined modalities from **PDL**, the formulas can be rendered "human readable," circumventing a standard critique of the inscrutability of μ-calculus notation.

Let $\mathfrak{M}_H$ be an LTS model of a hybrid automaton H. Since the H-reachability relation satisfies $h = (er)^*e$, the diamond and box modalities for h can be defined by

$$\langle \mathbf{h} \rangle \varphi \stackrel{\text{def}}{=} \mu Z.\langle \mathbf{e} \rangle \varphi \vee \langle \mathbf{e} \rangle \langle \mathbf{r} \rangle Z$$
$$[\mathbf{h}] \varphi \stackrel{\text{def}}{=} \nu Z.[\mathbf{e}] \varphi \wedge [\mathbf{e}][\mathbf{r}] Z$$

and thus

$$[\![\langle \mathbf{h} \rangle \varphi]\!]^{\mathfrak{M}_H} = \bigcup_{k \in \mathbb{N}} [\![\langle (\mathbf{er})^k \mathbf{e} \rangle \varphi]\!]^{\mathfrak{M}_H}$$
$$[\![[\mathbf{h}] \varphi]\!]^{\mathfrak{M}_H} = \bigcap_{k \in \mathbb{N}} [\![[(\mathbf{er})^k \mathbf{e}] \varphi]\!]^{\mathfrak{M}_H}.$$

1) Invariance: $[\mathbf{h}]\varphi$ denotes the largest H-invariant set contained in the set of φ-states, provided $\varphi \rightarrow \mathbf{Inv}$ (where $\mathbf{Inv} \leftrightarrow \langle \mathbf{e} \rangle \mathbf{tt} \leftrightarrow \langle \mathbf{h} \rangle \mathbf{tt}$) and hence $[\mathbf{h}]\varphi \rightarrow \varphi$; in general, $[\mathbf{h}]\varphi$ denotes the largest H-invariant set contained in the set of $[\mathbf{e}]\varphi$-states, the latter also including all of $\neg \mathbf{Inv}$.

2) Safety: $\psi \rightarrow [\mathbf{h}]\varphi$ is true in $\mathfrak{M}_H$ iff every H-trajectory that starts in a ψ-state always remains in the set of φ-states.

3) Reachability: $\langle \breve{\mathbf{h}} \rangle \psi$ denotes the region H-reachable from the set of ψ-states, where $\langle \breve{\mathbf{h}} \rangle \varphi \stackrel{\text{def}}{=} \mu Z.\langle \breve{\mathbf{e}} \rangle \varphi \vee \langle \breve{\mathbf{e}} \rangle \langle \breve{\mathbf{r}} \rangle Z$. Applying the axioms $Ax.\langle \breve{a} \rangle$-1 and $Ax.\langle \breve{a} \rangle$-2, the formulas $\langle \breve{\mathbf{h}} \rangle \psi \rightarrow \varphi$ and $\psi \rightarrow [\mathbf{h}]\varphi$ are equivalent.

4) Jump-Infinite Liveness: Every maximal H-trajectory starting in a ψ-state makes infinitely many discrete jumps, iff the sentence

$$\psi \rightarrow [\mathbf{h}]\langle \mathbf{e} \rangle \mathbf{Grd}$$
$$\wedge \bigwedge_{q \in Q} \left((\mathbf{Inv}_q \rightarrow \langle \mathbf{f}_q \rangle \neg \mathbf{Inv}_q) \right.$$
$$\left. \wedge \bigwedge_{q' \colon (q,\, q') \in E} \left(\mathbf{Grd}_{q,\, q'} \wedge \langle \breve{\mathbf{h}} \rangle \psi \right) \rightarrow [\mathbf{r}_{q,\, q'}] \mathbf{Inv}_{q'} \right)$$

is true in $\mathfrak{M}_H$. The first conjunct says that from all states H-reachable from ψ-states, it is possible to evolve into one of the guard sets, and the second says that for each q, the flow ϕ_q will eventually leave Inv_q, and all resets from that part of $Grd_{q,q'}$ that is H-reachable from ψ must lead to $Inv_{q'}$.

5) Inevitability: $\psi \rightarrow [\mathbf{h}]\langle \mathbf{h} \rangle \varphi$ says that from every state H-reachable from a ψ-state, there is a H-trajectory leading to a φ-state.

6) Eventuality: $\psi \rightarrow [\mathbf{h}](\varphi_1 \rightarrow \langle \mathbf{h} \rangle \varphi_2)$ says that for every H-trajectory from a ψ-state, if it ever reaches a φ_1-state, then it has a further extension that eventually reaches a φ_2-state.

7) Non-Zeno Liveness: A sufficient, but not necessary, condition for the non-Zenoness of all H-trajectories starting in a ψ-state is the existence of a $\Delta > 0$ such that

$$\bigwedge_{(q'',q) \in E} \left(\left(\mathbf{Grd}_{q'',q} \wedge \langle \breve{\mathbf{h}} \rangle \psi \right) \rightarrow \right.$$
$$\left. \bigwedge_{q' \colon (q,q') \in E} [\mathbf{r}_{q'',q}] \left(\langle \mathbf{e}_q \rangle \mathbf{Grd}_{q,q'} \rightarrow \neg \left\langle \mathbf{e}_q^{<\Delta} \right\rangle \mathbf{Grd}_{q,q'} \right) \right)$$

where $\mathbf{e}_q^{<\Delta}$ is the (strict) Δ-upper-bounded evolution relation.

8) Instances of Stability: For *fixed* $\varepsilon < 0$ and $\delta > 0$, define $(\boldsymbol{\delta})^{\mathfrak{M}_H} = \{((q, \mathbf{x}), (q', \mathbf{x}') | q' = q \wedge d(\mathbf{x}, \mathbf{x}') < \delta\}$ and likewise for $(\boldsymbol{\varepsilon})^{\mathfrak{M}_H}$. Then the sentence

$$\varphi \rightarrow [\mathbf{h}]\varphi \wedge \langle \boldsymbol{\delta} \rangle \varphi \rightarrow [\mathbf{h}]\langle \boldsymbol{\varepsilon} \rangle \varphi$$

says that φ is a H-invariant set, and that every H-trajectory that starts within δ of φ always remains within ε of φ.

With propositional variables ranging over all subsets of the state space, we have the machinery to *compare* relations. The relational inclusion $a^{\mathfrak{M}} \subseteq b^{\mathfrak{M}}$ holds iff $\mathfrak{M} \models \langle a \rangle Z \rightarrow \langle b \rangle Z$ iff $\mathfrak{M} \models [b]Z \rightarrow [a]Z$, i.e., for all possible assignments of a set $A \subseteq S$ to the free variable Z.

Comparison of relations allows us to formalize questions of *approximate verification* in $\mathbf{L}\mu$. Suppose we wish to verify a safety sentence in a system H with complex nonlinear dynamics, for which model checking is not possible (discussed in Section V-A). One then seeks out a simpler system H' that is an *overapproximation* of H, and for which model checking is possible. For example, H' could be chosen so that each of the component evolution relations satisfy $e_q \subseteq e'_q$, and the reset relations are the same, so $h \subseteq h'$, and hence $[\mathbf{h}']Z \rightarrow [\mathbf{h}]Z$. Then the approximate safety sentence $\psi \rightarrow [\mathbf{h}']\varphi$ entails the desired safety sentence $\psi \rightarrow [\mathbf{h}]\varphi$. However, if model checking returned the answer that $\psi \rightarrow [\mathbf{h}']\varphi$ is not true, so $[\![\psi \wedge \langle \mathbf{h}' \rangle \neg \varphi]\!]^{\mathfrak{M}} \neq \emptyset$, then the implication $\langle \mathbf{h} \rangle Z \rightarrow \langle \mathbf{h}' \rangle Z$ does not allow us to conclude anything about $\psi \wedge \langle \mathbf{h} \rangle \neg \varphi$. The metric tolerance relations can also be used to formalize notions of *tightness* of approximations.

The high-level idea of *robustness* is that for a given *nominal model* H, and an *uncertainty class* $\mathfrak{H}$ of models H' that are possible *variations* of H in some well-quantified respects, one wants to ascertain whether each of the models H' possess the same qualitative or quantitative properties as H [25]. As one approach to formalizing robustness, consider a class $\mathfrak{H}$ consisting of hybrid automata H' that differ from H in at most their evolution relations; for some fixed

$\varepsilon > 0$, suppose the evolution relations satisfy $e'_q \subseteq e_q \circ B_\varepsilon$ for each $q \in Q$. This says that if γ' is an integral curve of ϕ'_q witnessing an evolution $x \xrightarrow{e'_q} y$ in a variant H', then there is an integral curve γ of ϕ_q starting from x, witnessing $x \xrightarrow{e_q} y$ in the nominal system H, such that γ' lies within a *ε-tube* around γ (see [25] and [47], which consider ε-tubes around time sequences). Translating this relational inclusion into modal formulas, we have

$$[\mathbf{e}][\varepsilon]Z \rightarrow [\mathbf{e}']Z$$

and hence

$$[\mathbf{h}_\varepsilon]\, Z \rightarrow [\mathbf{h}']Z$$

where

$$[\mathbf{h}_\varepsilon]\, Z \stackrel{\text{def}}{=} \nu W.[\mathbf{e}][\varepsilon]Z \wedge [\mathbf{e}][\varepsilon][\mathbf{r}]W$$
$$[\mathbf{h}']\, Z \stackrel{\text{def}}{=} \nu W.\, [\mathbf{e}']\, Z \wedge [\mathbf{e}']\, [\mathbf{r}]W.$$

So for the verification of a *robust safety* property, it would suffice to prove $\psi \rightarrow [\mathbf{h}_\varepsilon]\varphi$, since this entails $\psi \rightarrow [\mathbf{h}']\varphi$ for any such variant H'. One can also work with the finitary versions of the hybrid reachability relations restricted to some bounded finite number of resets.

G. Temporal Logics for Hybrid and Timed Systems

There are numerous proposals in the literature for the extension to hybrid and timed systems of temporal logics developed for discrete systems. Our focus here is on extensions of temporal logics, which specifically address issues that arise from working with real time, including the expression of *quantitative* temporal requirements.

1) Branching Temporal Logics: Real-time extensions of branching temporal logics include *Timed* **CTL (TCTL)** [53], *Integrator* **CTL (ICTL)**, and Timed μ-calculus (**TLμ**) [20], [21]. These logics were developed for reasoning about quite restricted classes of hybrid automata: *timed automata*, in which all real-valued coordinates are *clocks*, with dynamics $\dot{x}_i = 1$, and (so-called) *linear hybrid automata*, with straight line flows $\phi_q(\mathbf{x}, t) = \mathbf{x} + t\mathbf{u}$ for some slope or rate vector $\mathbf{u} \in \mathbb{R}^n$; for both classes, all reset relations and distinguished subsets are restricted to those definable by Boolean combinations of inequalities $x_i + c \leq x_j + d$, for constants $c, d \in \mathbb{N}$. However, their semantics can be cleanly extended to arbitrary hybrid automata in the manner developed here.

The first issue addressed is the appropriate semantics for the *until* operators. Intuitively, we would like a state $s \in S_H$ to satisfy $\varphi_1(\exists \mathbf{h}\mathcal{U})\varphi_2$ iff there is *some* H-trajectory η from s along which there is an η-successor state s' at which φ_2 is satisfied, and φ_1 is continuously satisfied at all η-intermediate states between s and s'. In standard **CTL**, $\varphi_1(\exists\mathcal{U})\varphi_2$ is characterized by the μ-calculus translation $\mu Z.\varphi_2 \vee (\varphi_1 \wedge \exists \bigcirc Z)$, with iteration of the "next-step" operator $\exists\bigcirc$. Over LTS models of hybrid automata, where $\exists\bigcirc$ is $\langle \mathbf{e} \cup \mathbf{r}\rangle$, this fails to capture the intended meaning of φ_1 holding continuously. The key to the solution in [20], [21] is a means to refer to *regions* of the state space through which a trajectory may pass. We reformulate this idea in our modal framework.

Any subset $A \subseteq S_H = S$ of hybrid states has a unique decomposition $A = \bigcup_{q\in Q}(\{q\} \times A_q)$. Then for each control mode $q \in Q$, define the relativized *A-evolution* relation $e_q(A)$: $S \rightsquigarrow S$ by

$$(l, \mathbf{x}) \xrightarrow{e_q(A)} (l', \mathbf{x}') \stackrel{\text{def}}{\Leftrightarrow} l' = l = q \wedge \mathbf{x} \xrightarrow{e(A_q \cap Inv_q,\, \phi_q)} \mathbf{x}'$$

and for each discrete transition $(q, q') \in E$, define the relativized *A-reset* relation $r_{q,q'}(A)$: $S \rightsquigarrow S$ by

$$(l, \mathbf{x}) \xrightarrow{r_{q,q'}(A)} (l', \mathbf{x}') \stackrel{\text{def}}{\Leftrightarrow} l = q \wedge l' = q' \wedge \mathbf{x} \xrightarrow{R_{q,q'}} \mathbf{x}' \wedge \mathbf{x} \in A_q \wedge \mathbf{x}' \in A_{q'}.$$

For any μ-calculus sentence $\varphi \in \mathcal{S}_\mu(\Sigma_H, \Phi_H)$, define relations $\mathbf{e}(\varphi)^{\mathfrak{M}_H} \stackrel{\text{def}}{=} \cup_{q\in Q} e_q([\![\varphi]\!]^{\mathfrak{M}_H})$ and $\mathbf{r}(\varphi)^{\mathfrak{M}_H} \stackrel{\text{def}}{=} \cup_{(q,\,q')\in E} r_{q,\,q'}([\![\varphi]\!]^{\mathfrak{M}_H})$. So $\mathbf{e}(\mathbf{tt})$ is just $\mathbf{e}$ and likewise $\mathbf{r}(\mathbf{tt})$ is just $\mathbf{r}$. Then define

$$\varphi_1(\exists \mathbf{h}\mathcal{U})\varphi_2 \stackrel{\text{def}}{=} \mu Z.\langle\, \mathbf{e}\,(\varphi_1)\rangle\varphi_2 \vee \langle \mathbf{e}\,(\varphi_1)\rangle\langle \mathbf{r}\,(\varphi_1)\rangle Z.$$

Hence, $\langle \mathbf{h}\rangle\varphi \leftrightarrow \mathbf{tt}(\exists \mathbf{h}\mathcal{U})\varphi$, as one would expect. The characterization of $(\exists\mathcal{U})$ in [20] and [21] replaces φ_1 with $\varphi_1 \vee \varphi_2$ in the body of the μ-calculus formula because they want to retain the implication $\varphi_2 \rightarrow \varphi_1(\exists\mathcal{U})\varphi_2$; for the definition here, we have instead $(\varphi_1 \wedge \varphi_2) \rightarrow \varphi_1(\exists \mathbf{h}\mathcal{U})\ \varphi_2$, and $\varphi_1(\exists \mathbf{h}\mathcal{U})\varphi_2 \rightarrow \langle \mathbf{h}\rangle(\varphi_1 \wedge \varphi_2)$.

The hybrid *all-until* construct, which expresses notions of *inevitability*, is a yet more complicated creature, and we do not give a detailed treatment here. Intuitively, $\varphi_1(\forall \mathbf{h}\mathcal{U})\varphi_2$ is satisfied at a state $s \in S_H$ if along *every maximal* H-trajectory η starting from s, *there is* an η-successor s' at which φ_2 is satisfied, and φ_1 is satisfied at all η-intermediate states between s and s'. For example, the *event sequence requirement* in our example control problem in Section II-D is formalized by the sentence

$$\bigwedge_{k\in K} (\mathbf{A}_k \wedge \mathbf{Good}) \rightarrow \mathbf{A}_k(\forall\, \mathbf{h}\mathcal{U})\mathbf{A}_{k\ominus 1}.$$

In [21], an $(\forall\mathcal{U})$ operator for hybrid trajectories is shown to be μ-calculus definable, using $(\exists\mathcal{U})$ and time-bounded all-until operators $(\forall\mathcal{U})_{\leq c}$.

Quantitative time-bounded properties can be formalized by extending the branching-time languages with *specification clocks*, which are additional real-valued variables distinct from system coordinate variables. If H is a hybrid automata over state space $S \subseteq Q \times \mathbb{R}^n$, then formulas in the enriched language containing m specification clock variables are interpreted in an expanded LTS $\mathfrak{M}_H^m$ model over $S \times (\mathbb{R}^+)^m$. Specification clocks z have the continuous dynamics $\dot{z} = 1$ in all control modes q, and remain constant under reset relations. The additional constructs of the language are atomic clock constraints, of the form $z_1 + c \leq z_2 + d$ with constants $c, d \in \mathbb{N}$, which can be treated as extra atomic propositions, and the "*freeze*" or "*clock-reset*" construct $z.\varphi$, which corresponds to the action of starting a timer from zero. The formula $z.\varphi$ is satisfied at all extended

states $s \in S \times (\mathbb{R}^+)^m$ such that $s(z := 0)$ satisfies φ. For example, the **TCTL** formula $\forall\Box z.(\psi \Rightarrow \forall\Diamond(\varphi \wedge z \leq 17))$ asserts that along all trajectories, a ψ-state is followed by a φ-state within 17 time units.

2) Linear Temporal Logics: Formulas of linear temporal logics are interpreted with respect to *execution sequences* or *traces* of transition system models (Definition 6). An **LTL** formula in true in $\mathfrak{M}$ if it is satisfied by all infinite execution sequences starting from the designated set of initial states of $\mathfrak{M}$. This semantics can be related to the state-based semantics by working in a derived LTS model $\mathfrak{M}^\omega$ whose state space is a suitable set of ω-length execution sequences or traces of $\mathfrak{M}$ [33].

Much of the work on timed and hybrid extensions of linear temporal logics has concentrated on the theory of *timed ω-words*, extending the rich relationship between (untimed) ω-languages, formulas of **LTL**, and Büchi automata over ω-words [65], [77]. Over an arbitrary LTS model $\mathfrak{M}$, a *timed execution sequence* is a pair (σ, τ), where $\sigma = \langle s_i, a_i\rangle_{i\in\omega}$ is an infinite execution sequence of $\mathfrak{M}$, and $\tau = \langle t_i\rangle_{i\in\omega}$ is an infinite sequence of positive reals $t_i \in \mathbb{R}^+$, interpreted as the *delay* between the successive states s_i and s_{i+1} under the transition $s_i \xrightarrow{a_i^{\mathfrak{M}}} s_{i+1}$. A *timed trace* (π, τ) is defined similarly. Logics such as *metric temporal logic* **MTL** [13] are obtained by extending **LTL** with (integer endpoint) interval-bounded versions of the *until*, *always,* and *sometimes* temporal operators, and formulas are interpreted over timed execution sequences or timed traces. For example, $\varphi_1\mathcal{U}_{[3,\,17]}\varphi_2$ is read "φ_1 holds until φ_2 does, and that happens in between 3 and 17 time units." A similar extension of **LTL** with time-bounded temporal operators is developed in [83]. The survey paper [77] examines the decidability and complexity of model checking for several variants of **MTL** with respect to LTS models of timed automata.

3) Interval Temporal Logics: Hybrid temporal logic (**HTL**) [78], [79] *extended duration calculus* (**EDC**) of [15] and [28] are both first-order temporal logics that replace states as instantaneous valuations of variables, with state *functions* over an interval of time. In [78] and [79], the basic semantic objects are *phases* $(I, \overline{f})$, where $I = [c, d] \subseteq \mathbb{R}^+$ (or $I = [c, d)$) is a time interval, and $\overline{f}$ is a vector of type-consistent functions of time $f_v\colon I \to \mathbb{R}$ and $f_v\colon I \to Q$ that are piece-wise continuous (or smooth) if variable v is real-valued, and piecewise-constant for discrete v. Both **HTL** and **EDC** have a "*chop*" operator, from which the *until*, *always*, and *sometimes* temporal operators are definable and include a means to refer to the duration for which a formula has been satisfied.

H. Expressing Topological and Continuity Properties

Work on temporal logics for hybrid systems has focussed on the metric aspects of *real time*. Here, we turn our attention to the metric and more general *topological* structure of *real space*. We have already seen how within the framework of (plain) modal logic **PML**, we can reason about some metric structure using modalities $\langle\boldsymbol{\delta}\rangle$ and $[\boldsymbol{\delta}]$ for concrete δ-tolerance relations $B_\delta\colon S \rightsquigarrow S$ on metric spaces S, and use these to formalize some notions of stability and robustness. In [23] and [50], we show how modal logic also provides a means to represent a *topology* on the state space of an LTS or Kripke model.

Formally, we extend the syntax of **PML** or **L**μ to include an additional "plain" or "unlabeled" box modality $\Box$, and its dual diamond $\Diamond$, with $\Diamond\varphi \stackrel{\text{def}}{=} \neg\Box\neg\varphi$. A *topological LTS model* $\mathfrak{M} = (S, \mathcal{T}, \{a^{\mathfrak{M}}\}_{a\in\Sigma}, \{[\![p]\!]^{\mathfrak{M}}\}_{p\in\Phi})$ is an LTS model in which $(S, \mathcal{T})$ is a *topological space*. From Tarski and McKinsey [84], the axioms for the box modality $\Box$ of the well-studied modal logic **S4** correspond exactly to the Kuratowski axioms for the topological interior operator $int_{\mathcal{T}}$, and dually the **S4** diamond corresponds to the topological closure $cl_{\mathcal{T}}$. The additional semantic clauses for the extended language interpreted in topological LTS models are then

$$[\![\Diamond\varphi]\!]^{\mathfrak{M}}_\xi \stackrel{\text{def}}{=} cl_{\mathcal{T}}\left([\![\varphi]\!]^{\mathfrak{M}}_\xi\right)$$
$$[\![\Box\varphi]\!]^{\mathfrak{M}}_\xi \stackrel{\text{def}}{=} int_{\mathcal{T}}\left([\![\varphi]\!]^{\mathfrak{M}}_\xi\right).$$

Call the resulting logics **TopPML** and **TopL**μ (the prefix **T** already being used for *timed* extensions of temporal logics). The **S4** axioms for $\Box$ are as follows:

$$\begin{aligned} Ax.\Box\wedge\colon &\quad \Box(Z \wedge W) \leftrightarrow (\Box Z \wedge \Box W)\\ Ax.\Box\mathbf{tt}\colon &\quad \Box\mathbf{tt} \leftrightarrow \mathbf{tt}\\ Ax.\Box\mathbf{T}\colon &\quad \Box Z \to Z\\ Ax.\Box\mathbf{4}\colon &\quad \Box Z \to \Box\Box Z. \end{aligned}$$

In the enriched language, we can simply express topological properties of sets of states. A sentence φ denotes an *open* (*closed*) set in $(S, \mathcal{T})$ iff $\varphi \to \Box\varphi$ ($\Diamond\varphi \to \varphi$) is true in $\mathfrak{M}$. The sentence $\Diamond\varphi \wedge \neg\Box\varphi$ denotes the topological *boundary* $bd_{\mathcal{T}}([\![\varphi]\!]^{\mathfrak{M}})$.

We now have the resources with which to formalize notions of *continuity*. In purely topological terms, a single-valued function $f\colon (X, \mathcal{T}) \to (Y, \mathcal{U})$ is *continuous* if for every open set U in $(Y, \mathcal{U})$, the inverse-image $f^{-1}(U)$ is open in $(X, \mathcal{T})$. The corresponding notions for set-valued maps were introduced by Kuratowski and Bouligand in the 1930s and use the preimage operators instead of the inverse image [57], [58], [60].

Definition 15: A relation $r\colon (X, \mathcal{T}) \rightsquigarrow (Y, \mathcal{U})$ is said to be *lower semicontinuous* (*lsc*) if for every open set U in $(Y, \mathcal{U})$, the preimage $\mathrm{Pre}^{\exists}(r)(U)$ is open in $(X, \mathcal{T})$; r is said to be *upper semicontinuous* (*usc*) if for every open set U in $(Y, \mathcal{U})$, the preimage $\mathrm{Pre}^{\forall}(r)(U)$ is open in $(X, \mathcal{T})$, or equivalently, for every closed set C in $(Y, \mathcal{U})$, the set $\mathrm{Pre}^{\exists}(r)(C)$ is closed in $(X, \mathcal{T})$; and r is called *continuous* if it is both usc and lsc.

For a relation $a^{\mathfrak{M}}\colon S \rightsquigarrow S$ in a topological LTS model $\mathfrak{M}$, the semicontinuity properties are simply expressed by the formulas

$$\begin{aligned} \text{lsc-}a\colon &\quad \langle a\rangle\Box Z \to \Box\langle a\rangle Z\\ \text{usc-}a\colon &\quad [a]\Box Z \to \Box[a]Z. \end{aligned}$$

The lsc-a formula asserts that for every subset $A \subseteq S$, the inclusion $\mathrm{Pre}^{\exists}(a^{\mathfrak{M}})(int_{\mathcal{T}}(A)) \subseteq int_{\mathcal{T}}(\mathrm{Pre}^{\exists}(a^{\mathfrak{M}})(A))$ holds, and this is equivalent to the lsc property for $a^{\mathfrak{M}}$; likewise for the usc property. From these simple modal characterizations, we can give a purely formal proof within the axiomatic proof system that each of the semicontinuity properties is inherited under finite compositions and finite unions of relations. For the infinitary Kleene star operation, if a is lsc then a^* is lsc, but the corresponding result for usc relations does not hold in general (basically because the infinite intersection of a family of open sets need not be open).

We briefly mention two lines of enquiry opened up by consideration of semi-continuity properties of relations. The first confirms our interest in *finite topologies* as combinatorial structures, which shed some light on *discretizations* of continuous and hybrid spaces, and notions of *stability* for such discretizations. From [51], an LTS model $\mathfrak{M}$ has a *bisimulation equivalence* of finite index exactly when there is a finite topology $\mathcal{T}$ on S such that each of the transition relations $a^{\mathfrak{M}}\colon S \rightsquigarrow S$ are *continuous* with respect $\mathcal{T}$, and each of the atomic sets $[\![p]\!]^{\mathfrak{M}}$ is *clopen* (both closed and open) in $\mathcal{T}$. The partition determined by an equivalence relation gives the extreme case where all open sets are also closed, so $\mathcal{T} = Clop(\mathcal{T})$ is actually a Boolean algebra.

Returning to the example in Section II-D, the cover $\{A_{k,m}\}_{(k,m)\in P} \cup \{A_0\}$ generates a finite topology $\mathcal{T}_P$ on $X := \mathbb{R}^2$. As it stands, this is not a *subtopology* of the standard topology on $\mathbb{R}^2$ (c.f. [46]), since $Safe_{k,c}$ will be open-intersect-closed, but with an extra $\Box$ operator in the modal characterization, it could be made so. As noted in [46], violations in continuity conditions for *finite* topologies can be finitarily detected. Looking at Fig. 5, we can see that for the set $[\![\mathbf{A}_{2,4} \wedge \mathbf{A}_{1,2}]\!]^{\mathfrak{M}}$ in $\mathcal{T}_P$, its $\exists$-pre-image $[\![\langle \mathbf{e}_{(2,4),\mathbf{ac}}\rangle(\mathbf{A}_{2,4} \wedge \mathbf{A}_{1,2})]\!]^{\mathfrak{M}}$, which cuts across $A_{2,4}$, is not in $\mathcal{T}_P$. One can iteratively construct a finer topology $\mathcal{T} \supseteq \mathcal{T}_P$ with respect to which each of the relations $e_{(k,m),c}$ for $((k,m),c) \in Q$ are both lsc and usc by adding more preimage sets.

The second line of enquiry looks at the semicontinuity properties with respect to the standard topology as primitive forms of *metric stability*. From [57] and [58], under the hypotheses that X is a compact metric space and the set image $r(x)$ is a closed for each $x \in X$, the semicontinuity properties have an a ε-δ characterization: a relation $r\colon X \rightsquigarrow X$ is usc iff for all $x \in X$ and all $\varepsilon > 0$, there is a $\delta > 0$ such that for all x', $y' \in X$

$$\begin{aligned} &d_X(x, x') < \delta \text{ and } x' \xrightarrow{r} y' \\ \Rightarrow &(\exists y \in X)\left[x \xrightarrow{r} y \text{ and } d_Y(y, y') < \varepsilon\right]. \end{aligned}$$

In words, if x' is close to x then the set image $r(x')$ is *nearly contained in* $r(x)$, in the sense that $r(x')$ is contained in the ε-ball or *tube* around $r(x)$, as illustrated in Fig. 8 for set-images $r(x)$ that are curves from x. Similarly, $r\colon X \rightsquigarrow X$ is lsc iff whenever x' is close to x then the set-image $r(x')$ *nearly contains* $r(x)$, in the sense that $r(x)$ is contained in the ε-ball around $r(x')$.

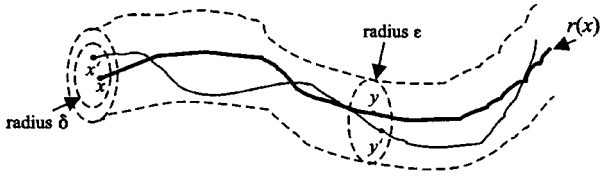


Fig. 8. The usc property in the compact metric setting.

For a hybrid automaton H, consider the bounded-jump versions $h_N\colon S_H \rightsquigarrow S_H$ of the H-reachability relation h given by $h_N = \bigcup_{k\le N}(er)^k e$. So $h_N(q, \mathbf{x})$ is the set of all states lying on some H-trajectory from $(q, \mathbf{x})$ that has at most N discrete jumps. Let $h_N^{\#}\colon X \rightsquigarrow X$ and $h^{\#}\colon X \rightsquigarrow X$ be the projections of h_N and h onto $X \subseteq \mathbb{R}^n$. Then under the hypotheses required for the metric ε-δ characterizations, the semicontinuity properties for $h_N^{\#}$ and $h^{\#}$ express elementary notions of stability of hybrid trajectories. A preliminary study of the usc property for hybrid reachability relations is given in [23].

V. Methods and Tools for Formal Verification and Synthesis

A. Symbolic Model Checking

Model checking for all standard propositional modal and temporal logics proceeds by first giving a translation into the μ-calculus. The task is to compute the set $[\![\varphi]\!]^{\mathfrak{M}} \subseteq S$, and so determine whether $\mathfrak{M} \models \varphi$. The high-level algorithm for computing $[\![\varphi]\!]^{\mathfrak{M}}$ for a μ-calculus sentence is just the inductive definition of the formal semantics in Definitions 11 and 13, recursively breaking a sentence into its subsentences. For fixed-point sentences $\mu Z.\varphi$, one calls a recursive subprocedure that computes the approximation sequence $[\![\varphi^k]\!]^{\mathfrak{M}}$ and returns the answer $[\![\mu Z.\varphi]\!]^{\mathfrak{M}} = [\![\varphi^k]\!]^{\mathfrak{M}}$ if the sequence converges at stage k.

The question then becomes: For which classes of LTS models $\mathcal{K}$ is it the case that the high-level algorithm for model checking $\mathbf{L}\mu$ sentences is a) *effectively implementable* and b) guaranteed to *terminate* in a finite number of steps on all inputs $(\mathfrak{M}, \varphi)$?

The question would not be asked if one were only interested in *finite* LTS models. Indeed, a good measure of the success of formal methods for discrete systems can be attributed to the tractability of model checking over finite LTS models. For example, for **CTL** and the small fragment of $\mathbf{L}\mu$ needed to capture **CTL**, there are model-checking algorithms of time and space complexity $O(\|\mathfrak{M}\| \cdot |\varphi|)$; for the full μ-calculus, the time complexity is $NP \cap \text{co-}NP$ [37]. *Binary decision diagrams* (BDDs) provide an efficient means to represent finite sets of states, and the Boolean and modal operations on them, and have been successfully used for model checking systems with upwards of 10^{100} states [37].

For general classes of LTS models, a primary means of addressing the issue b) of finite termination is via Proposition 14. It suffices to identify classes $\mathcal{K}$ of LTS models, all of which have a finite bisimulation quotient. As surveyed in [31], finite bisimulation results have been established, on the one hand, for the restricted class of *timed automata* [24] and

some extensions, and on the other, for the mathematically richer class of *o-minimal hybrid automata* [85], whose component flows ϕ_q, sets Inv_q and $Grd_{q,q'}$, and resets $R_{q,q'}$ in $X \subseteq \mathbb{R}^n$ are all first-order definable in an *o-minimal* (*model-theoretic*) *structure* $\mathcal{R} = (\mathbb{R}; <, +, -, 0, 1, \cdots)$ expanding the reals $\mathbb{R}$ as an ordered Abelian group, but subject to the further restriction that the reset maps $R_{q,q'}$ are *set-valued constant* (see also [51]). O-minimality is statable as a syntactic condition on first-order definability, but its core content is a *topological finiteness property*: every set $A \subseteq \mathbb{R}^m$ first-order definable in an o-minimal structure has only finitely many connected components [86]. The class of o-minimal structures over $\mathbb{R}$ is quite rich. It includes the structure $\mathcal{RCF} = (\mathbb{R}; <, +, -, \cdot, 0, 1)$ of $\mathbb{R}$ as a *real closed field*; the quantifier-free first-order formulas in the language $\mathcal{L}(\mathcal{RCF}) = \{<, +, -, \cdot, 0, 1\}$ are Boolean combinations of equalities and inequalities of *polynomials* $P(x_1, \cdots, x_m)$, and the sets $A \subseteq \mathbb{R}^m$ so definable are called *semialgebraic* sets. The class of o-minimal structures also includes the richer structure $\mathcal{RCF}_{\exp}$ obtained by adding the exponential function; the structure $\mathcal{RCF}_{\text{r.a.}}$ obtained by adding finitely many analytic functions restricted to a bounded rectangle; their combination $\mathcal{RCF}_{\exp,\,\text{r.a.}}$; and yet further expansions [85], [86].

Proof of finite termination is only half the question. For the issue a) of an effective implementation, one needs a finitary syntactic means of representing sets of states that is closed under the Boolean and preimage operators, and furthermore, that representation must be *decidable* in the sense that it can be determined by finite computation whether distinct representations are semantically equal. This requires that there be an *effectively representable* and *decidable* modal algebra $\mathcal{B} \subseteq \mathcal{P}(S)$ for $\mathfrak{M}$ such that $\mathcal{B}(\mathfrak{M}) \subseteq \mathcal{B}$ (see [20] and [38]).

Of the o-minimal structures over $\mathbb{R}$, the richest known to have the required effectiveness and decidability is $\mathcal{RCF}$. By the famous Tarski–Seidenberg results, there is an algorithm that transforms any first-order formula in the language $\mathcal{L}(\mathcal{RCF})$ into a quantifier-free formula in $QF(\mathcal{RCF})$ that is equivalent over $\mathcal{RCF}$, so proving that all sets first-order definable in $\mathcal{RCF}$ are semialgebraic, and furthermore, the semialgebraic sets are decidable. In contrast, the richer structure $\mathcal{RCF}_{\exp}$ does not have the quantifier elimination property (although it does have the weaker but significant property called *model completeness*), and the decidability of its first-order theory is still an open question.

For LTS models of hybrid automata, it suffices to consider the underlying LTS model over $S := X \subseteq \mathbb{R}^n$. Let $\theta_S(v_1, \cdots, v_n)$, $\{\rho_a(v_1, \cdots, v_n, v_1', \cdots, v_n')\}_{a\in\Sigma}$, and $\{\theta_p(v_1, \cdots, v_n)\}_{p\in\Phi}$ be a list of formulas in $QF(\mathcal{RCF})$ defining the components of $\mathfrak{M}$. The following recursive translation map takes as input an $\mathbf{L}\mu$ sentence $\varphi \in \mathcal{S}_\mu(\Sigma, \Phi)$, and if it terminates, it returns $|\varphi| \in QF(\mathcal{RCF})$

$$
\begin{aligned}
|p| &= \theta_p(v_1, \cdots, v_n) \\
|\neg\varphi| &= \theta_S(v_1, \cdots, v_n) \wedge \neg|\varphi| \\
|\varphi_1 \vee \varphi_2| &= |\varphi_1| \vee |\varphi_2| \\
|\langle a\rangle\varphi| &= \text{QE}\lim((\exists v_1') \cdots (\exists v_n') : \\
&\qquad \rho_a(v_1, \cdots, v_n, v_1', \cdots, v_n') \wedge \\
&\qquad |\varphi|\,[v_1 := v_1', \cdots, v_n := v_n']) \\
|\mu Z.\varphi| &= |\varphi^k| \text{ for the least } k \text{ such that} \\
&\qquad \mathcal{RCF} \models |\varphi^k| \leftrightarrow |\varphi^{k+1}|\,.
\end{aligned}
$$

Note that termination is not problematic for **PML** sentences. Applying the best available algorithm by Basu *et al.* [87] (which significantly improves the version of Collins' cylindrical algebraic decomposition currently implemented in the computer algebra tool REDLOG [88]), the number of arithmetical operations required to perform this QElim procedure is bounded by $s^{(n+1)^2} d^{(n+1)O(n)}$, when the body of the argument formula is defined by a total of s polynomials in n variables, each of at most degree d. Checking equivalence or emptiness (unsatisfiability) of formulas in $QF(\mathcal{RCF})$ has a bound of $s^{n+1} d^{O(n)}$. The translation also extends to topological LTS models and sentences of topological extensions of **PML** and $\mathbf{L}\mu$, where the topology $\mathcal{T}$ is the standard metric topology from $\mathbb{R}^n$; this is because $int(A)$ is semialgebraic whenever A is semialgebraic, with a computable description in $QF(\mathcal{RCF})$ [86].

In our synthesis procedure illustrated by the example in Section II-D, each of the computations and decisions required in the course of the construction can be formulated as a model checking task for finitary **PML** sentences [32]. The procedure can thus be effectively implemented for instances of Problem 4 where the input data of the flows ϕ_c for $c \in C$ and the specification sets Bad and A_k are all semialgebraic. Our explicit semialgebraic description of the set $A_{2,4}$ is an example of the output of quantifier elimination. By applying [89], this sort of procedure can also be used when the continuous dynamics are given by certain classes of linear differential equations whose flows contain some exponential terms but for which the flow preimage $[\![\langle \mathbf{f}\rangle\varphi]\!]^{\mathfrak{M}}$ is semialgebraic whenever $[\![\varphi]\!]^{\mathfrak{M}}$ is semi-algebraic. Note that the presence of the exact reset maps $R_{q,q'} := test.Grd_{q,q'}$ as well as the metric tolerance relations B_δ mean that the LTS model of the final hybrid automaton is unlikely to have a finite quotient that meets the bisimulation conditions for *all* of the relations of the model, but this does not pose a problem if one is not model checking infinitary fixed-point sentences.

The model checking tool HYTECH [90] is designed for the restricted class of linear or polyhedral hybrid automata, all of whose real components are first-order definable by Boolean combinations of equalities and inequalities of linear terms $c_0 + c_1x_1 + \cdots + c_mx_m$. Rather than use quantifier elimination, a later version of HYTECH represents state sets in $\mathbb{R}^n$ as finite unions of *convex polyhedra*, given a vertex representation, and the Boolean operations and preimage operator are implemented using a library of standard polyhedral operations [21]. The work in [41] in this special issue describes an algorithm for *approximate* reachability and safety analysis of hybrid automata with linear differential equations, where the state sets are represented as special kinds of convex polyhedra, and applies this technology to a class of synthesis problems. The model checking tools KRONOS [91], COSPAN [92], and UPPAAL [93], all for the restricted

class of timed automata, represent convex data regions in $\mathbb{R}^n$ by integer-valued matrices, with operations performed using standard matrix operations.

B. Deductive Proof Systems

Deductive methods for verification and analysis differ from model checking in their methodology, their scope, and their degree of automation. Whereas model checking seeks to translate a specification formula into the lower level system description logic, or some other symbolic representation, deductive methods work by directly manipulating formulas in the high-level temporal or modal specification logic. Model checking can be implemented as a completely automated analysis tool, but its scope is restricted to *sentences* of the μ-calculus (or its sublogics or extensions) and to LTS models whose components have a sufficiently tractable first-order description. In contrast, deductive methods have a broader scope. They are applicable to all *formulas* of the specification logic, including such $\mathbf{L}\mu$ and $\mathbf{TopL}\mu$ formulas with free $Z \in \mathrm{PVar}$ as those expressing relational comparison and continuity properties. Moreover, the inferences of a proof system are valid over the largest class of formal models for which that system is *sound*, often the universal class of *all* models. Deductive methods can be used not just for the verification of single properties, considered one at a time, but also for the larger enterprise of building up a *deductive theory* or *knowledge base* of formulas true in a model. The flip side is that implementations of deductive verification methods tend to only be semiautomated, typically combining some degree of automated proof search, user-interactive proof construction, and automated proof checking.

The Hilbert-style proof system for $\mathbf{L}\mu$ is due to Kozen [36]. On top of the axioms and rules for **PML**, this proof system for $\mathbf{L}\mu$ has the fixed-point axiom

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

and the inference rules

$$Rule.\text{Subst.:}\quad \frac{\varphi}{\varphi[Z := \psi]}$$

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

For formulas $\varphi \in \mathcal{F}_\mu(\Sigma, \Phi)$, we write $\vdash_{\mathbf{L}\mu} \varphi$ if there is a formal proof of φ in this proof system for $\mathbf{L}\mu$ and say φ is a *theorem* of $\mathbf{L}\mu$. The *soundness* of this proof system is quite straightforward. This says, of all formulas $\varphi \in \mathcal{F}_\mu(\Sigma, \Phi)$, if $\vdash_{\mathbf{L}\mu} \varphi$ then φ is universally *valid*, meaning $\mathfrak{M} \models \varphi$ for *all* LTS models $\mathfrak{M}$, of arbitrary cardinality and character. The *validity problem* for $\mathbf{L}\mu$, of determining whether a formula $\varphi \in \mathcal{F}_\mu(\Sigma, \Phi)$ is valid, is EXPTIME complete [33]. More recently, the *completeness* of Kozen's axiomatization has been established, namely, that if φ is valid, then $\vdash_{\mathbf{L}\mu} \varphi$. A drawback of the proof of this result in [94] is that it does not extend in any modular fashion to axiomatic extensions, such as that for $\mathbf{TopL}\mu$, which adds the **S4** axioms for $\Box$ (Section IV-H).

In trying to establish that $\mathfrak{M} \models \varphi$ using a proof system for $\mathbf{L}\mu$ or $\mathbf{TopL}\mu$, one seeks to show that φ is a *deductive consequence* of a list Γ of formulas ψ such that $\mathfrak{M} \models \psi$ is already known. In selecting Γ, one has at one's disposal a hierarchy of formulas that form a *knowledge base* about $\mathfrak{M}$:

- formulas that are theorems of the proof system, that are true in all LTS models;
- formulas that are true in all models in the intended class, such as the LTS models of hybrid automata; these arise as deductive consequences of definitions in $\mathbf{L}\mu$ of modal/temporal operators like $[\mathbf{h}]$ and $(\exists \mathbf{h}\mathcal{U})$, and formula schemes such as $\langle \mathbf{f} \rangle \mathbf{T}$, $\langle \mathbf{f} \rangle \mathbf{4}$ and $\langle \mathbf{f} \rangle \mathbf{L}$ (Section IV-C) for symbols $\mathbf{f}$ for the orbit relations of flows;
- formulas that have already been directly verified as true in $\mathfrak{M}$, either deductively, or for sentences, perhaps by a call to a separate model checking tool.

To see a proof system in action, we give a formal proof in $\mathbf{TopL}\mu$ that if a relation $a^{\mathfrak{M}}$: $S \rightsquigarrow S$ is lsc, then so is its Kleene star

1.	$\langle a \rangle \Box Z \rightarrow \Box \langle a \rangle Z$	Assumption
2.	$\langle a \rangle \langle a^* \rangle Z \rightarrow \langle a^* \rangle Z$	**PDL** theorem
3.	$\Box \langle a \rangle \langle a^* \rangle Z \rightarrow \Box \langle a^* \rangle Z$	2.: *Rule*.Mono
4.	$\langle a \rangle \Box \langle a^* \rangle Z \rightarrow \Box \langle a \rangle \langle a^* \rangle Z$	1.: *Rule*.Subst
5.	$\langle a \rangle \Box \langle a^* \rangle Z \rightarrow \Box \langle a^* \rangle Z$	4., 3.: **PL**
6.	$Z \rightarrow \langle a^* \rangle Z$	**PDL** theorem
7.	$\Box Z \rightarrow \Box \langle a^* \rangle Z$	6.: *Rule*.Mono
8.	$(\Box Z \vee \langle a \rangle \Box \langle a^* \rangle Z) \rightarrow \Box \langle a^* \rangle Z$	5., 7.: **PL**
9.	$\mu W.(\Box Z \vee \langle a \rangle W) \rightarrow \Box \langle a^* \rangle Z$	8.: *Rule*.μ-f.p
10.	$\langle a^* \rangle \Box Z \rightarrow \Box \langle a^* \rangle Z$	9.: Def. $\langle a^* \rangle$.

Rule.Mono is the derived rule for any modal operator O (box or diamond): from $\psi \rightarrow \varphi$, infer $O\psi \rightarrow O\varphi$, and **PL** is propositional logic.

From the $\mathbf{L}\mu$ characterization of $[\mathbf{h}]$ together with the dual inference rule *Rule*.ν-f.p in $\mathbf{L}\mu$, one can readily derive an obvious *invariance induction rule* for proving safety properties of hybrid automata

$$Rule.\text{Inv-}\mathbf{h}\text{:}\quad \frac{\begin{array}{ll} \psi \rightarrow \chi & \\ \chi \rightarrow [\mathbf{e}_q]\chi & \text{for each } q \in Q \\ \chi \rightarrow [\mathbf{r}_{q,q'}]\chi & \text{for each } (q, q') \in E \\ \chi \rightarrow \varphi & \end{array}}{\psi \rightarrow [\mathbf{h}]\varphi.}$$

The premises of the rule assert that the set of χ-states is *invariant* under both evolution and reset relations, and that χ is *intermediate* between ψ and φ. This is a cleaner μ-calculus analog of the invariance rule in **LTL**-based logics used for the verification of safety properties for hybrid automata in [12]–[14] and implemented in the verification tool STeP [95].

In the course of our controller synthesis construction in Section II-D, we generate a list of **PML** formulas that are true in a model $\mathfrak{M}_C$ over the state space $S := X \subseteq \mathbb{R}^n$, using

a suitable model checking tool; a partial list is given in Section IV-D. The idea is that the *synthesis* is *self-verifying*, in the sense that the correctness of construction can be demonstrated by formally deducing the $\mathbf{L}\mu$ formulas encoding performance specifications (given in Sections IV-F and IV-G) from this list of **PML** formulas. For ease of discussion, we continue to work in $\mathfrak{M}_C$ rather than shift the analysis up to the hybrid model $\mathfrak{M}_H$ over $S := Q \times X$.

For the *safety property* $\mathbf{Good} \rightarrow [\mathbf{h}]\neg\mathbf{Bad}$, we use $Rule.$Inv-$\mathbf{h}$. We only need to check $\mathbf{Good} \rightarrow [\mathbf{r}_{q,q'}]\mathbf{Good}$, since the initiality condition $\mathbf{Good} \rightarrow \neg\mathbf{Bad}$ and the evolution invariance $\mathbf{Good} \rightarrow [\mathbf{e}_q]\mathbf{Good}$ are already established. For each $(q,q') = ((p,c),(p',c')) \in E$, the reset $R_{q,q'} = test.(A_p \cap A_{p'})$, so it follows that $[\mathbf{r}_{q,q'}]Z \leftrightarrow [(\mathbf{A}_p \wedge \mathbf{A}_{p'})?]Z \leftrightarrow ((\mathbf{A}_p \wedge \mathbf{A}_{p'}) \rightarrow Z)$. We then need to prove $\mathbf{Good} \rightarrow ((\mathbf{A}_p \wedge \mathbf{A}_{p'}) \rightarrow \mathbf{Good})$, but this is a propositional tautology, so we are done. The *jump-infinite liveness* property can also be formally verified using $Rule.$Inv-$\mathbf{h}$ to prove $\psi \rightarrow [\mathbf{h}]\langle\mathbf{e}\rangle\mathbf{Grd}$. For the *non-Zeno liveness* property, we need to calculate a lower bound Δ on the time duration between resets, which could be done using the semialgebraic descriptions of the cover sets $A_{k,m}$. For the *event sequence requirement* expressed using $(\forall\mathbf{h}\mathcal{U})$ (Section IV-G), the idea is that chains of local inevitability formulas such as $\mathbf{A}_{2,4} \rightarrow [\mathbf{e}_{(2,4),\mathbf{ac}}]\langle\mathbf{e}_{(2,4),\mathbf{ac}}\rangle(\mathbf{A}_1 \vee \mathbf{A}_{2,5})$ and $\mathbf{A}_{2,5} \rightarrow [\mathbf{e}_{(2,5),c}]\langle\mathbf{e}_{(2,5),c}\rangle\mathbf{A}_1$ will entail $(\mathbf{A}_2 \wedge \mathbf{Good}) \rightarrow \mathbf{A}_2(\forall\mathbf{h}\mathcal{U})\mathbf{A}_1$.

The example illustrates how inference rules like $Rule.$Inv-$\mathbf{h}$ can be used in *conjunction* with model checking tools. Similar ideas are developed in [14], using the tool STeP. This verification tool combines the model checking capability of automatically generating first-order translations of candidate invariant sentences χ, such as $\mathbf{Init} \vee \langle\dot{\mathbf{e}}\rangle\mathbf{tt} \vee \langle\dot{\mathbf{r}}\rangle\mathbf{tt}$, with proof systems for LTL-based logics. In related work, envisaged for discrete systems but more generally applicable, the general-purpose verification environment *prototype verification system* (PVS) [96] is used to give an integration of model checking and theorem proving; this is achieved by encoding the propositional μ-calculus within the (classical) simply typed higher order logic on which PVS is based.

C. Controller Synthesis

In our example in Section II-D, we demonstrate one way of formulating a controller synthesis problem for hybrid systems within our logic framework. The cover and AD map $\{A_p\}_{p\in P}$ was found using some custom-designed *predecessor* operators on sets, namely, $\mathrm{Pred}^{\delta\text{-in}}_{k,c}(Z) \stackrel{\mathrm{def}}{=} \mathbf{A}_k \wedge \langle\boldsymbol{\delta}\rangle\langle\mathbf{e}_{k,c}\rangle Z$ and $\mathrm{Pred}^{\mathrm{out}}_{k,c}(Z) \stackrel{\mathrm{def}}{=} \mathbf{A}_k \wedge [\mathbf{e}_{k,c}]\neg Z$, applied to $Z := \langle\boldsymbol{\delta}\rangle\mathbf{Bad}$. By the nature of that construction, it did not involve any iteration, but subject to the restriction on inclusion-monotonicity (which holds for $\mathrm{Pred}^{\delta\text{-in}}_{k,c}$, but not for $\mathrm{Pred}^{\mathrm{out}}_{k,c}$), we can iterate any set operator of our design.

Fixed points of operators on sets are a dominant theme in work on controller synthesis for hybrid systems, and for DES systems before them. Translating Ramadge and Wonham's *modular feedback logic* [39] into modal logic (and extending it from single-valued partial functions to arbitrary relations), the *maximal control invariant subset* of a set of states $P \subseteq S$ is obtained as the greatest fixed point of the operator

$$\mathrm{H}(Z) \stackrel{\mathrm{def}}{=} \mathbf{P} \wedge \left(\bigwedge_{a\in\Sigma_u} [a]Z \right)$$

where $\Sigma_u = \Sigma - \Sigma_c$ is the subalphabet of *uncontrollable* events and $\delta_a\colon S \rightsquigarrow S$ is the component relation for event $a \in \Sigma$ of the system transition relation $\delta\colon S \times \Sigma \rightsquigarrow S$. In words, $\mathrm{H}(Z)$ is the set of states s in P such that for each uncontrollable event $a \in \Sigma_u$, the a-successors of s are all in Z. Control is effected by a supervisor's being able to override the system transition relation and *disable* controllable events $a \in \Sigma_c$ at states $s \in \mathrm{dom}(\delta_a)$. A state *feedback supervisor* is a map $f\colon S \rightsquigarrow \Sigma$ such that $\Sigma_u \subseteq f(s)$ for all $s \in S$, and $a \in \Sigma_c$ is disabled by f at s if $a \notin f(s)$. From any control invariant set $P \subseteq S$, one can construct a state feedback supervisor whose application renders P an invariant set; the *maximal* control invariant subset of P then gives rise to the *least restrictive* supervisor to enforce the invariance of P. In [39], modularity is considered with respect to conjunction of predicates; working within $\mathbf{L}\mu$, a richer level of modularity is attainable.

The construction in [39] is specifically adapted to hybrid systems in [97], where the system model is essentially a hybrid automaton over state space $S = Q \times X$ and $X \subseteq \mathbb{R}^n$, with reset maps $r_a\colon S \rightsquigarrow S$ indexed by events $a \in \Sigma$, and for $a \in \Sigma_c$, a supervisor can override and disable a reset r_a at states $s \in Grd_a$. Earlier work on controller synthesis for timed automata in [98] is along the same general lines.

In work on controller synthesis in this special issue, [41] considers a class of control problems in which one starts with a complete hybrid automaton H, and the synthesis task is to find the largest *subsystem* $H' \preceq H$, in the sense that the state space and flows are the same, but $Inv'_q \subseteq Inv_q$ and $Grd'_{q,q'} \subseteq Grd_{q,q'}$ (with the resets always $R_{q,q'} := test.Grd_{q,q'}$), such that the subsystem H' satisfies a safety property. Their solution is a greatest fixed-point construction using a customized predecessor operator on subsets of hybrid states. Transcribed in modal logic, this operator is of the form

$$\pi(Z) \stackrel{\mathrm{def}}{=} Z \wedge \left[\left(\bigvee_{q\in Q} [\mathbf{e}_q]\, Z \right) \vee \left(\bigvee_{(q,\,q')\in E} \langle\mathbf{e}_q(Z)\rangle\, (\mathbf{Grd}_{q,\,q'} \wedge Z) \right) \right]$$

where $\mathbf{e}_q(Z)$ denotes the relativized evolution relation defined in Section IV-G, requiring the substitution $Z := \varphi$ of a sentence to give it concrete meaning. In words, s is in $\pi(Z)$ iff s is in Z, and either for some $q \in Q$, there is a q-evolution from s that remains in Z for all time, or else it is possible

to q-evolve from s for some time, while remaining within Z, and then switch and still be in Z.

The controller synthesis problems considered in [42] are more complex—their hybrid automata have continuous control and disturbance inputs, and the task is to construct a feedback control map (with both discrete and continuous values), which restricts the behavior of the system so as to satisfy a safety property. But there again, we see a greatest fixed-point construction of a maximal controlled invariant subset. Due to quantification over control and disturbance *functions* $u\colon I \to U$ and $d\colon I \to V$ on a time interval I (rather than *values* in U and V), their controllable and uncontrollable predecessor operators, and their two-place Reach operator, require reformulation to be expressible using modal operators. In earlier work in hybrid systems (with differing system models), fixed points of operators on sets were used in [99]–[101] to characterize the *viability kernel* of a set of states as the largest subset invariant under hybrid trajectories and from which all hybrid trajectories are jump-infinite.

In a separate development, generalizations of LTS models have been formulated for settings where the natural model is a *game* between two or more *agents*, with logics called *alternating temporal logic* and *alternating μ-calculus* developed for the specification of system properties [80]. The formalisms are general enough to cover the supervisory control framework of discrete event systems, as well module checking and receptiveness problems for open discrete and hybrid systems.

VI. Conclusion

The primary goal of this work was to seek out and render perspicuous unifying threads in the substantial literature on logics and formal methods for hybrid systems. The key to such a unification is to be found by appreciating the genericness of relational transition system models as abstract dynamical systems, and the power and simplicity of the μ -calculus as the "parent logic" for reasoning about them. A larger theme is how *old* resources—formal models, logics, proof systems, decision procedures—can be put to *new* uses in the formal analysis and synthesis of hybrid systems.

Of the many suggested in the text, several lines of future research warrant special mention.

First, given the demand for maximally powerful model checking tools, there is a pressing need for investigation of *efficient quantifier elimination* for the pre- or postimages of polynomial flows applied to semialgebraic sets, starting with a study of [87].

Given the intrinsic limits on algorithmic model checking, further investigation is required of the theory and practice of *approximating* systems with complex nonlinear dynamics by systems with tractable semialgebraic (or simpler) flows.

There is much more to be done in deductive methods for the μ-calculus and its extensions; in particular, *tableaux proof systems* offer the best available automated theorem provers [56]. Further work is also needed on clean architectures for combining model checking and deductive methods, building on [95], [96].

To get full value out of our modal representation of topological and metric structure, more study is needed of notions of stability and robustness for hybrid systems framed in the language of the general topology of relations and their integration with the concepts developed from classical control theory.

Given the natural occurrence of distributed, multiagent hybrid systems, there is a clear need for further investigation of logics for these systems, applying and building on [74] and [80].

Acknowledgment

The authors' view of hybrid systems and logics for reasoning about them has benefited from fruitful conversations and exchanges with many people; in particular, they would like to thank J. Remmel, W. Kohn, A. Yakhnis, S. Artemov, D. Kozen, D. Cook, J. Miller, K. Rudie, A. Walz, S. Sastry, G. Lafferriere, G. Pappas, J. Lygeros, M. Prandini, J. Hespanha, R. Goré, N. Bonnette, B. Anderson, T. Brinsmead, S. Dey, and T. Moor. They also thank X. Krump for excellent tutorials on computer graphics.

References

[1] R. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, Eds., *Hybrid Systems*: Springer-Verlag, 1993. LNCS 736.
[2] P. J. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, Eds., *Hybrid Systems II*: Springer-Verlag, 1995. LNCS 999.
[3] A. Pnueli and J. Sifakis, Eds., *Theoret. Comput. Sci.*, 1995, vol. 138. (Special Issue on Hybrid Systems).
[4] R. Alur, T. A. Henzinger, and E. D. Sontag, Eds., *Hybrid Systems III*: Springer-Verlag, 1996. LNCS 1066.
[5] O. Maler, Ed., *Hybrid and Real-Time Systems: International Workshop HART'97*: Springer-Verlag, 1997. LNCS 1201.
[6] P. J. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, Eds., *Hybrid Systems IV*: Springer-Verlag, 1997. LNCS 1273.
[7] P. J. Antsaklis and A. Nerode, Eds., *IEEE Trans. Automat. Contr.*, Apr. 1998, vol. 43. (Special Issue on Hybrid Control Systems).
[8] T. A. Henzinger and S. Sastry, Eds., *Hybrid Systems: Computation and Control (HSCC'98)*: Springer-Verlag, 1998. LNCS 1386.
[9] P. J. Antsaklis, W. Kohn, M. Lemmon, A. Nerode, and S. Sastry, Eds., *Hybrid Systems V*: Springer-Verlag, 1999. LNCS 1567.
[10] F. W. Vaandrager and J. H. van Schuppen, Eds., *Hybrid Systems: Computation and Control (HSCC'99)*: Springer-Verlag, 1999. LNCS 1569.
[11] E. M. Clarke and J. M. Wing, "Formal methods: State of the art and future directions," *ACM Comput. Surveys*, vol. 28, pp. 626–643, Dec. 1996. Rep. Strategic Directions in Computing Research Formal Methods Working Group, ACM Workshop, Aug. 1996.
[12] Z. Manna and A. Pnueli, "Verifying hybrid systems," in *Hybrid Systems*, R. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, Eds: Springer-Verlag, 1993, pp. 4–35. LNCS 736.
[13] ——, "Models for reactivity," *Acta Informatica*, vol. 30, pp. 609–678, 1993.
[14] Z. Manna and H. B. Sipma, "Deductive verification of hybrid systems using STeP," in *Hybrid Systems: Computation and Control (HSCC'98)*, T. A. Henzinger and S. Sastry, Eds: Springer-Verlag, 1998, pp. 305–318. LNCS 1386.
[15] Z. Chaochen, A. P. Ravn, and M. R. Hansen, "An extended duration calculus for hybrid real-time systems," in *Hybrid Systems*, R. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, Eds: Springer-Verlag, 1993, pp. 36–59. LNCS 736.
[16] L. Lamport, "Hybrid systems in TLA+," in *Hybrid Systems*, R. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, Eds: Springer-Verlag, 1993, pp. 77–102. LNCS 736.
[17] P. Herrmann, G. Graw, and H. Krumm, "Compositional specification and structured verification of hybrid systems in cTLA," in *Proc. 1st IEEE Int. Symp. Object-Oriented Real-Time Distributed Computing*, 1998, pp. 335–340.

[18] 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 *Hybrid Systems*, R. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, Eds: Springer-Verlag, 1993, pp. 209–229. LNCS 736.

[19] 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," *Theoret. Comput. Sci.*, vol. 138, pp. 3–34, 1995.

[20] T. A. Henzinger, "The theory of hybrid automata," in *Proc. 11th Annu. IEEE Symp. Logic Comput. Sci. (LICS'96)*, 1996, pp. 278–292.

[21] R. Alur, T. A. Henzinger, and P.-H. Ho, "Automatic symbolic verification of embedded systems," *IEEE Trans. Software Eng.*, vol. 22, pp. 181–201, 1996.

[22] Y. Zhang and A. K. Mackworth, "Constraint nets: A semantic model for hybrid dynamic systems," *Theoret. Comput. Sci.*, vol. 138, pp. 211–239, 1995.

[23] J. M. Davoren, "On hybrid systems and the modal μ-calculus," in *Hybrid Systems V*, P. J. Antsaklis, W. Kohn, M. Lemmon, A. Nerode, and S. Sastry, Eds: Springer-Verlag, 1999, pp. 38–69. LNCS 1567.

[24] R. Alur and D. L. Dill, "A theory of timed automata," *Theoret. Comput. Sci.*, vol. 126, pp. 183–235, 1994.

[25] C. Horn and P. J. Ramadge, "Robustness issues for hybrid systems," in *Proc. 34th Int. Conf. Decision Contr., CDC'95*, 1995, pp. 1467–1472.

[26] Y. Zhang and A. K. Mackworth, "Specification and verification of hybrid dynamic systems with timed $\forall$-automata," in *Hybrid Systems III*, R. Alur, T. A. Henzinger, and E. D. Sontag, Eds: Springer-Verlag, 1996, pp. 587–603. LNCS 1066.

[27] N. A. Lynch, R. Segala, F. Vaandrager, and H. B. Weinberg, "Hybrid I/O automata," in *Hybrid Systems III*, R. Alur, T. A. Henzinger, and E. D. Sontag, Eds: Springer-Verlag, 1996, pp. 496–510. LNCS 1066.

[28] Z. Chaochen, W. Ji, and A. P. Ravn, "A formal description of hybrid systems," in *Hybrid Systems III*, R. Alur, T. A. Henzinger, and E. D. Sontag, Eds: Springer-Verlag, 1996, pp. 511–530. LNCS 1066.

[29] M. Rönkkö and A. P. Ravn, "Actions systems with continuous behaviour," in *Hybrid Systems V*, P. J. Antsaklis, W. Kohn, M. Lemmon, A. Nerode, and S. Sastry, Eds: Springer-Verlag, 1999, pp. 304–323. LNCS 1567.

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

[31] R. Alur, T. A. Henzinger, G. Lafferriere, and G. Pappas, "Discrete abstractions of hybrid systems," *Proc. IEEE*, vol. 88, pp. xxx–xxx, July 2000.

[32] J. M. Davoren, "Using modal logics for the formal analysis and synthesis of hybrid control systems," Australian National University, Computer Sciences Laboratory, RSISE, Tech. Rep. TR-ARP-02-2000, Jan. 2000.

[33] C. Stirling, "Modal and temporal logics," in *Handbook of Logic in Computer Science*, S. Abramsky, D. M. Gabbay, and T. Maibaum, Eds. Oxford, U.K.: Oxford Univ. Press, Clarendon, 1992, vol. 2, pp. 477–563.

[34] A. Nerode and R. Shore, *Logic for Applications*, 2nd ed. Berlin, Germany: Springer-Verlag, 1997. Graduate Texts in Computer Science.

[35] R. Goldblatt, *Logics of Time and Computation*, 2nd ed. Stanford, CA: CLSI Publications, 1992.

[36] D. Kozen, "Results on the propositional μ-calculus," *Theoret. Comput. Sci.*, vol. 27, pp. 333–354, 1983.

[37] E. A. Emerson, "Model checking and the mu-calculus," in *Descriptive Complexity and Finite Models*, N. Immerman and P. G. Kolaitis, Eds: AMS, 1997, pp. 185–208.

[38] T. A. Henzinger and R. Majumdar, "A classification of symbolic transition systems," in *Proc. 17th Int. Symp. Theoret. Aspects of Computer Science (STACS'00)*: Springer-Verlag, 2000. LNCS.

[39] P. J. Ramadge and W. M. Wonham, "Modular feedback logic for discrete event systems," *SIAM J. Contr. Optim.*, vol. 25, pp. 1202–1218, 1987.

[40] P. J. Ramadge and W. M. Wonham, "The control of discrete event systems," *Proc. IEEE*, vol. 77, pp. 81–98, 1989.

[41] E. Asarin, O. Bournez, T. Dang, O. Maler, and A. Pnueli, "Effective synthesis of switching controllers for linear systems," *Proc. IEEE*, vol. 88, pp. 1011–1025, July 2000.

[42] C. J. Tomlin, J. Lygeros, and S. S. Sastry, "A game theoretic approach to controller design for hybrid systems," *Proc. IEEE*, vol. 88, pp. 949–970, July 2000.

[43] C. Livadas, J. Lygeros, and N. A. Lynch, "High-level modeling and analysis of the air traffic alert and collision avoidance system (TCAS)," *Proc. IEEE*, vol. 88, pp. 926–948, July 2000.

[44] R. De Carlo, M. Branicky, S. Pettersson, and B. Lennartson, "Perspectives and results on the stability and stabilizability of hybrid systems," *Proc. IEEE*, vol. 88, pp. 1069–1082, July 2000.

[45] X. D. Koutsoukos, P. J. Antsaklis, J. A. Stiver, and M. D. Lemmon, "Supervisory control of hybrid systems," *Proc. IEEE*, vol. 88, pp. 1026–1049, July 2000.

[46] A. Nerode and W. Kohn, "Models for hybrid systems: Automata, topologies, controllability, observability," in *Hybrid Systems*, R. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, Eds: Springer-Verlag, 1993, pp. 297–316. LNCS 736.

[47] V. Gupta, T. A. Henzinger, and R. Jagadeesan, "Robust timed automata," in *Hybrid and Real-Time Systems: International Workshop HART'97*, O. Maler, Ed: Springer-Verlag, 1997, pp. 331–345. LNCS 1201.

[48] R. E. Kalman, P. L. Falb, and M. A. Arbib, *Topics in Mathematical System Theory*. New York: McGraw-Hill, 1969.

[49] S. N. Artemov, J. M. Davoren, and A. Nerode, "Logic, topological semantics and hybrid systems," in *Proc. 36th Int. Conf. Decision Contr., CDC'97*, 1997, pp. 698–701.

[50] J. M. Davoren, "Modal Logics for Continuous Dynamics," Ph.D. thesis, Dept. Mathematics, Cornell Univ., Jan. 1998.

[51] ——, "Topologies, continuity and bisimulations," *Theoret. Inform. Applicat.*, vol. 33, pp. 357–381, 1999.

[52] J. Sifakis, "A unified approach for studying the properties of transition systems," *Theoret. Comput. Sci.*, vol. 18, pp. 227–258, 1982.

[53] R. Alur, C. Courcoubetis, and D. L. Dill, "Model-checking in dense real-time," *Inform. Computat.*, vol. 104, pp. 2–34, 1993.

[54] A. Deshpande, A. Göllü, and P. Varaiya, "SHIFT: A formalism and a programming language for dynamic networks of hybrid automata," in *Hybrid Systems IV*, P. J. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, Eds: Springer-Verlag, 1997, pp. 113–133. LNCS 1273.

[55] V. Gupta, R. Jagadeesan, and V. A. Saraswat, "`Hybrid cc`, hybrid automata and program verification," in *Hybrid Systems III*, R. Alur, T. A. Henzinger, and E. D. Sontag, Eds: Springer-Verlag, 1996, pp. 52–75. LNCS 1066.

[56] R. Goré, "Tableaux methods for modal and temporal logics," in *Handbook of Tableaux Methods*, M. D'Agistino *et al.*, Eds. Norwell, MA: Kluwer, 1999, pp. 297–396.

[57] J.-P. Aubin and H. Frankowska, *Set-Valued Analysis*. Boston, MA: Birkhäuser, 1990.

[58] E. Akin, *The General Topology of Dynamical Systems*. Providence, RI: American Mathematical Society, 1993.

[59] J. R. Munkres, *Topology: A First Course*. Englewood Cliffs, NJ: Prentice-Hall, 1975.

[60] C. Berge, *Topological Spaces, Including a Treatment of Multi-Valued Functions, Vector Spaces and Convexity*. New York: Dover, 1997.

[61] J. Palis and W. de Melo, *Geometric Theory of Dynamical Systems*. New York: Springer-Verlag, 1982.

[62] M. S. Branicky, "Topology of hybrid systems," in *Proc. 32nd Conf. Decision Contr.*, 1993, pp. 2309–2314.

[63] J. M. Davoren and T. Moor, "Logic-based design and synthesis of controllers for hybrid systems," Australian National Univ., Dept. Syst. Eng., RSISE, Canberra, Australia, Tech. Rep., July 2000.

[64] M. Y. Vardi and P. Wolper, "Automata-theoretic techniques in the modal logics of programs," *J. Comput. Syst. Sci.*, vol. 32, pp. 182–221, 1986.

[65] E. A. Emerson, "Temporal and modal logic," in *Handbook of Theoretical Computer Science*, J. van Leeuwen, Ed. Amsterdam: Elsevier, 1990, pp. 997–1072.

[66] J. G. Thistle and W. M. Wonham, "Control of infinite behavior of finite automata," *SIAM J. Contr. Optim.*, vol. 32, pp. 1075–1097, 1994.

[67] Y. Kesten, A. Pnueli, J. Sifakis, and S. Yovine, "Decidable integration graphs," in *Hybrid Systems*, R. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, Eds: Springer-Verlag, 1993, pp. 179–208. LNCS 736.

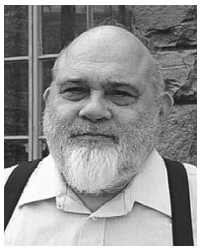
[68] E. W. Dijkstra, *A Discipline of Programming*. Englewood Cliffs, NJ: Prentice-Hall, 1976.

[69] D. D. Cook, "A study in the modeling, verification and control of hybrid systems," Ph.D. dissertation, Dept. Electrical and Electronic Engineering, Univ. of Melbourne, September 1999.

[70] R. Alur and T. A. Henzinger, "Modularity for timed and hybrid systems," in *CONCUR 97: Concurrency Theory*, A. Mazurkiewicz and J. Winkowski, Eds: Springer-Verlag, 1997, pp. 74–88. Lecture Notes in Computer Science.
[71] A. Pnueli, "The temporal logic of programs," in *Proc. 18th Annu. IEEE Symp. Foundations of Computer Science (FOCS'77)*, 1977, pp. 46–57.
[72] E. A. Emerson and E. M. Clarke, "Characterizing correctness properties of parallel programs as fixpoints," in *Proc. 7th Int. Coll. Automata, Languages and Programming*: Springer-Verlag, 1981, pp. 169–181. LNCS 85.
[73] S. Kripke, "Semantical analysis of modal logic I: Normal propositional calculi," *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, vol. 9, pp. 67–96, 1963.
[74] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi, *Reasoning About Knowledge*. Cambridge, MA: MIT Press, 1995.
[75] V. R. Pratt, "Semantical considerations on Floyd-Hoare logic," in *Proc. 17th Annu. IEEE Symp. Foundations of Computer Science (FOCS'76)*, 1976, pp. 109–121.
[76] D. Kozen and J. Tiuryn, "Logics of programs," in *Handbook of Theoretical Computer Science*, J. van Leeuwen, Ed. Amsterdam: Elsevier Science, 1990, pp. 789–840.
[77] T. A. Henzinger, "It's about time: Real-time logics reviewed," in *CONCUR 97: Concurrency Theory*, D. Sangiorgi and R. de Simone, Eds: Springer-Verlag, 1998, pp. 439–454. LNCS 1466.
[78] T. A. Henzinger, Z. Manna, and A. Pnueli, "Toward refining temporal specifications into hybrid systems," in *Hybrid Systems*, R. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, Eds: Springer-Verlag, 1993, pp. 60–76. LNCS 736.
[79] A. Kapur, T. A. Henzinger, Z. Manna, and A. Pnueli, "Proving safety properties of hybrid systems," in *FTRTFT 94: Formal Techniques in Real-time and Fault-tolerant Systems*, H. Langmaack, W.-P. de Roever, and J. Vytopil, Eds: Springer-Verlag, 1994, pp. 431–454. LNCS 863.
[80] R. Alur, T. A. Henzinger, and O. Kupferman, "Alternating-time temporal logic," in *Proc. 38th Annu. Symp. Foundations of Computer Science*, 1997, pp. 100–109.
[81] A. Tarski, "On the calculus of relations," *J. Symbolic Logic*, vol. 6, pp. 73–89, 1941.
[82] B. Jónsson and A. Tarski, "Boolean algebras with operators, part I," *Amer. J. Math.*, vol. 73, pp. 891–939, 1951.
[83] Y. Zhang and A. K. Mackworth, "Synthesis of hybrid constraint-based controllers," in *Hybrid Systems III*, R. Alur, T. A. Henzinger, and E. D. Sontag, Eds: Springer-Verlag, 1996, pp. 552–567. LNCS 1066.
[84] J. C. C. McKinsey and A. Tarski, "The algebra of topology," *Ann. Math.*, vol. 45, pp. 141–191, 1944.
[85] G. Lafferriere, G. J. Pappas, and S. Sastry, "O-minimal hybrid systems," *Math. Contr., Signals, Syst.*, vol. 13, pp. 1–21, 2000.
[86] L. van den Dries, *Tame Topology and O-minimal Structures*. Cambridge, U.K.: Cambridge Univ. Press, 1998. London Math. Soc. Lecture Notes 2483.
[87] S. Basu, R. Pollack, and M.-F. Roy, "On the combinatorial and algebraic complexity of quantifier elimination," *J. ACM*, vol. 43, pp. 1002–1045, 1996.
[88] A. Dolzmann, T. Sturm, and V. Weispfenning, "Real quantifier elimination in practice," in *Algorithmic Algebra and Number Theory*, B. Matzat, G. Greuel, and G. Hiss, Eds. Berlin, Germany: Springer-Verlag, 1998, pp. 221–248.
[89] G. Lafferriere, G. J. Pappas, and S. Yovine, "A new class of decidable hybrid systems," in *Hybrid Systems: Computation and Control (HSCC'99)*, F. W. Vaandrager and J. H. van Schuppen, Eds: Springer-Verlag, 1999, pp. 137–151. LNCS 1569.
[90] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi, "A user guide to HyTech," in *TACAS 95: Tools and Algorithms for the Construction and Analysis of Systems*, E. Brinksma, W. R. Cleaveland, K. G. Larsen, T. Margaria, and B. Steffen, Eds: Springer-Verlag, 1995, pp. 41–71. LNCS 1019.
[91] C. Daws, A. Olivero, S. Tripakis, and S. Yovine, "The tool KRONOS," in *Hybrid Systems III*, R. Alur, T. A. Henzinger, and E. D. Sontag, Eds: Springer-Verlag, 1996, pp. 208–219. LNCS 1066.
[92] R. Alur and R. P. Kurshan, "Timing analysis in COSPAN," in *Hybrid Systems III*, R. Alur, T. A. Henzinger, and E. D. Sontag, Eds: Springer-Verlag, 1996, pp. 220–231. LNCS 1066.
[93] J. Bengtsson, K. G. Larsen, F. Larsson, P. Pettersson, and W. Yi, "UPPAAL: A tool suite for automatic verification of real-time systems," in *Hybrid Systems III*, R. Alur, T. A. Henzinger, and E. D. Sontag, Eds: Springer-Verlag, 1996, pp. 232–243. LNCS 1066.
[94] I. Walukiewicz, "A note on the completeness of Kozen's axiomatization of the propositional μ-calculus," *Bull. Symbolic Logic*, vol. 2, pp. 349–366, 1996.
[95] Z. Manna, N. Bjorner, and A. Browne *et al.*, "An update on STeP: Deductive-algorithmic verification of reactive systems," in *Tool Support for System Specification, Development and Verification*: Springer-Verlag, 1998. LNCS.
[96] S. Owre, S. Rajan, J. M. Rushby, N. Shankar, and M. K. Srivas, "PVS: Combining specification, proof checking, and model checking," in *Computer-Aided Verification, CAV'96*: Springer-Verlag, 1996, pp. 411–414. LNCS 1102.
[97] H. Chen and H.-M. Hanisch, "Control synthesis of hybrid systems based on predicate invariance," in *Hybrid Systems V*, P. J. Antsaklis, W. Kohn, M. Lemmon, A. Nerode, and S. Sastry, Eds: Springer-Verlag, 1999, pp. 1–15. LNCS 1567.
[98] O. Maler, A. Pnueli, and J. Sifakis, "On the synthesis of discrete controllers for timed systems," in *Proceedings of STACS'95*, E. W. Mayr and C. Puech, Eds: Springer-Verlag, 1995, pp. 229–242. LNCS 900.
[99] A. Nerode, J. B. Remmel, and A. Yahknis, "Controllers as fixed-points of set-valued operators," in *Hybrid Systems II*, P. J. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, Eds: Springer-Verlag, 1995, pp. 344–358. LNCS 999.
[100] W. Kohn, A. Nerode, J. B. Remmel, and A. Yahknis, "Viability in hybrid systems," *Theoret. Comput. Sci.*, vol. 138, pp. 141–168, 1995.
[101] A. Deshpande and P. Varaiya, "Viable control of hybrid systems," in *Hybrid Systems II*, P. J. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, Eds: Springer-Verlag, 1995, pp. 128–147. LNCS 999.

J. M. Davoren (Member, IEEE) received the B.A. (Hons.) degree in mathematics and philosphy from the University of Melbourne, Victoria, Australia, in 1991, and the M.S. degree in computer science and the Ph.D. degree in mathematics from Cornell University, Ithaca, NY, in 1994 and 1998, respectively.

From 1998 to 1999, she was a Postdoctoral Fellow at the Center for Foundations of Intelligent Systems at Cornell University, with a visiting position in the Department of Electrical Engineering and Computer Science at the University of California, Berkeley, from February to April 1999. Since September 1999, she has been a Research Fellow at the Computer Sciences Laboratory in the Research School of Information Sciences and Engineering, Australian National University, Canberra, Australia. Her research interests lie in the fields of mathematical logic, theoretical computer science, systems and control theory, and general topology.



Anil Nerode (Member, IEEE) received the Ph.D. degree in mathematics from the University of Chicago, Chicago, IL, in 1956.

He was an NSF Postdoctoral Fellow with K. Gödel at the Institute for Advanced Studies (1957–1958) and a Visiting Assistant Professor at the University of California, Berkeley, with A. Tarski (1958–1959). In 1959, he joined the Faculty of Cornell University, Ithaca, NY, as an Assistant Professor in Mathematics, and served as the Director of the Center for Applied Mathematics from 1964 to 1965. He was also Chairman of the Department of Mathematics from 1982 to 1987 and the Director of the Mathematical Sciences Institute from 1987 to 1996. He is currently the Director of the Center for Foundations of Intelligent Systems and the Goldwin Smith Professor of Mathematics and Computer Science. His research interests include mathematical logic, computer science, and control engineering. He has published more than 150 papers and several books, and a summary of his research up to 1993 appears in J. N. Crossley *et al.* (Eds.), *Logical Methods: A Symposium in Honor of Anil Nerode's 60th Birthday* (Birkhäuser: Boston, MA, 1993).