

# Universal Computation and Other Capabilities of Hybrid and Continuous Dynamical Systems\*

Michael S. Branicky<sup>†</sup>

Laboratory for Information and Decision Systems  
and Center for Intelligent Control Systems  
Massachusetts Institute of Technology  
Cambridge, MA 02139

## Abstract

We explore the simulation and computational capabilities of hybrid and continuous dynamical systems. The continuous dynamical systems considered are ordinary differential equations (ODEs). For hybrid systems we concentrate on models that combine ODEs and discrete dynamics (e.g., finite automata). We review and compare four such models from the literature. Notions of simulation of a discrete dynamical system by a continuous one are developed. We show that hybrid systems whose equations can describe a precise binary timing pulse (exact clock) can simulate arbitrary reversible discrete dynamical systems defined on closed subsets of  $\mathbb{R}^n$ . The simulations require continuous ODEs in  $\mathbb{R}^{2n}$  with the exact clock as input. All four hybrid systems models studied here can implement exact clocks. We also prove that any discrete dynamical system in  $\mathbb{Z}^n$  can be simulated by continuous ODEs in  $\mathbb{R}^{2n+1}$ . We use this to show that smooth ODEs in  $\mathbb{R}^3$  can simulate arbitrary Turing machines, and hence possess the power of universal computation. We use the famous asynchronous arbiter problem to distinguish between hybrid and continuous dynamical systems. We prove that one cannot build an arbiter with devices described by a system of Lipschitz ODEs. On the other hand, all four hybrid systems models considered can implement arbiters even if their ODEs are Lipschitz.

**Key Words:** hybrid systems, universal computation, asynchronous arbiter, simulation, Turing machines, differential equations, dynamical systems

---

\*Work supported by the Army Research Office and the Center for Intelligent Control Systems under grants DAAL03-92-G-0164 and DAAL03-92-G-0115.

<sup>†</sup>Dept. of Electrical Engineering and Computer Science. Direct correspondence to: LIDS, MIT, 77 Mass. Ave., 35-415, Cambridge, MA 02139-4307. E-mail: [branicky@lids.mit.edu](mailto:branicky@lids.mit.edu)

# 1 Introduction

Hybrid systems are systems that combine both discrete and continuous dynamics. The continuous dynamics are usually represented by ordinary differential equations (ODEs). The discrete dynamics are generally governed by finite automata. The two interact at “event times” determined by the hitting of certain prescribed sets in the continuous state space.

In this paper, we explore the simulation and computational capabilities of hybrid systems. We concentrate on four models of hybrid systems in the control and dynamical systems literature [34, 3, 29, 12]. In each case, these models combine ODEs with some form of discrete dynamics. This paper is a step towards the characterization of these models in terms of the types of systems that can be described by, or “implemented” with, their equations. By construction, however, each model can implement ODEs with continuous vector fields (continuous ODEs). Thus, even with no discrete dynamics, these models can describe a large variety of phenomena.

In addition to “implementing” ODEs, all four models can implement a precise binary timing pulse or “exact clock” (defined later). Thus, we explore the capabilities of systems with continuous ODEs and exact clocks. For instance, we show such systems can simulate arbitrary reversible discrete dynamical systems defined on closed subsets of  $\mathbb{R}^n$ . These simulations require ODEs in  $\mathbb{R}^{2n}$  which use an exact clock as input.

Later, we find that one can still simulate arbitrary discrete dynamical systems defined on subsets of  $\mathbb{Z}^n$  without the capability of implementing an exact clock: one can use an approximation to an exact clock. Such an “inexact clock” is implemented with continuous functions of the state of a one-dimensional continuous ODE. As a result, one can perform such simulations using continuous ODEs in  $\mathbb{R}^{2n+1}$ . Turning to computational abilities, we show that continuous ODEs in  $\mathbb{R}^3$  possess the ability to simulate arbitrary Turing machines, pushdown automata, and finite automata. By simulating a universal Turing machine, we conclude that there exist ODEs in  $\mathbb{R}^3$  with continuous vector fields possessing the power of universal computation. Further, the ODEs simulating these machines may be taken smooth and do not require the machines to be reversible (cf. [26]).

Finally, we show that hybrid dynamical systems are strictly more powerful than Lipschitz ODEs in the types of systems they can implement. For this, we use a nontrivial example: the famous asynchronous arbiter problem [8, 23, 36]. First we quickly review the problem. Then we settle it in an ODE framework by showing that one cannot build an arbiter out of devices modeled by Lipschitz ODEs. Next, we examine the problem in a hybrid systems framework. We show that each of the four hybrid systems models can implement an arbiter even if their continuous dynamics is a system of Lipschitz ODEs.

The paper is organized as follows. In the next section we review some definitions from dynamical systems and develop some notation. In Section 3 we review four models of hybrid systems as presented in [34, 3, 29, 12]. We also briefly compare the different models. In Section 4 notions of simulation are discussed. Here, we make precise what we mean by “simulation” of discrete dynamical systems by continuous dynamical systems. All our simulation results are collected in Section 5. Section 6 deals with the asynchronous arbiter problem. The Appendix collects some technical lemmas.

## 2 Preliminaries

Throughout, we assume familiarity with standard notions and notations of analysis and topology [30, 28, 27]. Let  $\mathbb{R}$ ,  $\mathbb{R}^+$ ,  $\mathbb{Z}$ , and  $\mathbb{Z}^+$  denote the reals, nonnegative reals, integers, and nonnegative integers, respectively.

First, we review some standard definitions from dynamical systems [31, 17].

**Definition (Dynamical system)** *A continuous (resp. discrete) dynamical system defined on the topological space  $X$  over the semigroup  $\mathbb{R}^+$  (resp.  $\mathbb{Z}^+$ ) is a function  $f : X \times \mathbb{R}^+ \rightarrow X$  (resp.  $f : X \times \mathbb{Z}^+ \rightarrow X$ ) with the following three properties:*

1. *Initial condition:  $f(p, 0) = p$  for any point  $p \in X$ .*
2. *Continuity on both arguments.*
3. *Semigroup property:*

$$f(f(p, t_1), t_2) = f(p, t_1 + t_2),$$

*for any point  $p \in X$  and any  $t_1$  and  $t_2$  in  $\mathbb{R}^+$  (resp.  $\mathbb{Z}^+$ ).*

Technically, such functions are referred to as semi-dynamical systems, with the term dynamical system reserved for those which the semigroups  $\mathbb{R}^+$ ,  $\mathbb{Z}^+$  above may be replaced by the groups  $\mathbb{R}$ ,  $\mathbb{Z}$ . However, the more “popular” notion of dynamical system in math and engineering—and the one used here—requires only the semigroup property [15, 21]. Thus, the term *reversible* dynamical system is used when it is necessary to distinguish the group from semigroup case [22].

The shorthand  $[X, S, f]$  denotes the dynamical system  $f$  defined on  $X$  over the semigroup  $S$ ;  $X$  is referred to as its *state space* and points in  $X$  are called *states*. If a dynamical system is defined on a subset of  $X$ , we say it is a dynamical system *in*  $X$ .

For every fixed value of the parameter  $s$ , the function  $f(\cdot, s)$  defines a mapping of the space  $X$  into itself. Given  $[X, \mathbb{Z}^+, f]$ ,  $f(\cdot, 1)$  is its *transition function*. Thus if  $[X, \mathbb{Z}^+, f]$  is reversible, its transition function is invertible, with inverse given by  $f(\cdot, -1)$ .

The set  $f(p, S) = \{f(p, i) : i \in S\}$  is called the *orbit* or *trajectory of the point  $p$* . A *fixed point* of  $[X, S, f]$  is a point  $p$  such that  $f(p, s) = p$  for all  $s \in S$ . A set  $A \subset X$  is *invariant with respect to  $f$* , or simply *invariant*, if  $f(A, s) \subset A$  for all  $s \in S$ .

The notions of equivalence and homomorphism are crucial. Two dynamical systems  $[X, S, f]$ ,  $[Y, S, g]$  are said to be *isomorphic* (also *topologically equivalent* or simply *equivalent*) if there exists a homeomorphism  $\psi : X \rightarrow Y$  such that

$$\psi(f(p, s)) = g(\psi(p), s),$$

for all  $p \in X$  and  $s \in S$ . If the mapping  $\psi$  is only continuous, then  $[X, S, f]$  is said to be *homomorphic* to  $[Y, S, g]$ . Homomorphisms preserve trajectories, fixed points, and invariant sets.

In this paper, the continuous dynamical systems dealt with are defined by the solutions of ordinary differential equations (ODEs) [18]:

$$\dot{x}(t) = f(x(t)), \tag{2.1}$$

where  $x(t) \in X \subset \mathbb{R}^n$ . The function  $f : X \rightarrow \mathbb{R}^n$  is called a *vector field* on  $\mathbb{R}^n$ . The resulting dynamical system is then given by  $\phi(x_0, t) = x(t)$  where  $x(\cdot)$  is the solution

to Equation (2.1) starting at  $x_0$  at  $t = 0$ . (We assume existence and uniqueness of solutions; see [18] for conditions.) A system of ODEs is called *autonomous* or *time-invariant* if its vector field does not depend explicitly on time. Throughout, the shorthand *continuous* (resp. *Lipschitz*) *ODEs* denotes ODEs with continuous (resp. Lipschitz) vector fields.

An ODE *with inputs and outputs* [21, 33] is given by

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t)), \\ y(t) &= h(x(t)), \end{aligned} \tag{2.2}$$

where  $x(t) \in X \subset \mathbb{R}^n$ ,  $u(t) \in U \subset \mathbb{R}^m$ ,  $y \in Y \subset \mathbb{R}^p$ ,  $f : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$ , and  $h : \mathbb{R}^n \rightarrow \mathbb{R}^p$ . The functions  $u(\cdot)$  and  $y(\cdot)$  are the *inputs* and *outputs*, respectively.

Other notation is common [30, 27]:  $X \setminus U$  represents the complement of  $U$  in  $X$ ;  $\bar{U}$  represents the closure of  $U$ ,  $\partial U$  its boundary;  $f(t^+)$ ,  $f(t^-)$  denote the right-hand and left-hand limits of the function  $f$  at  $t$ , respectively; a function is right-continuous if  $f(t^+) = f(t)$  for all  $t$ ; unless specified  $\|x\|$  denotes an arbitrary norm of vector  $x$ ,  $\|x\|_2$  its Euclidean norm; the infinity norm of  $x \in \mathbb{R}^n$ , denoted  $\|x\|_\infty$ , is  $\max_{i=1}^n |x_i|$ .

Finally, for  $x \in \mathbb{R}$ ,  $\lfloor x \rfloor$  denotes the greatest integer less than or equal to  $x$ , and, in an abuse of common notation,  $\lceil x \rceil$  denotes the least integer greater than  $x$ ;  $|A|$ ,  $A$  a set, denotes its cardinality;  $A \simeq \{1, \dots, N\}$  means  $A$  is a set with  $|A| = N$ .

### 3 Models of Hybrid Systems

This section summarizes four precise models of hybrid systems developed from the control and dynamical systems point of view. For sure, there are many others and no review is attempted here [1, 16]. These have been chosen as much for the clarity and rigor of their presentation as for the mechanisms they use to combine discrete and continuous dynamics. Specifically, in Sections 3.1–3.4 we review the following models of hybrid systems, in order of (original) appearance of the cited papers:

1. Tavernini’s model [34],
2. Back-Guckenheimer-Myers model [3],
3. Nerode-Kohn model [29],
4. Brockett’s model [12].

Only the models are given here with minimal discussion. For further discussion and examples, the reader is referred to the original papers.

Some models in the papers above allow time-varying vector fields, but we only consider autonomous ones here. Also, we have sometimes changed notation from the original papers to make the presentation more uniform.

In Section 3.5, we briefly compare the four models.

#### 3.1 Tavernini’s Model

Tavernini discusses so-called *differential automata* in [34]. He was motivated to study such systems as a means of modeling phenomena with hysteresis.

A *differential automaton*,  $A$ , is a triple  $(S, f, \nu)$  where  $S$  is the state space of  $A$ ,  $S = \mathbb{R}^n \times Q$ ,  $Q \simeq \{1, \dots, N\}$  is the *discrete state space* of  $A$ , and  $\mathbb{R}^n$  is the *continuous state space* of  $A$ ;  $f$  is a finite family  $f(\cdot, q) : \mathbb{R}^n \rightarrow \mathbb{R}^n$ ,  $q \in Q$ , of vector fields, the *continuous dynamics* of  $A$ ; and  $\nu : S \rightarrow Q$  is the *discrete transition function* of  $A$ .

Let  $\nu_q \equiv \nu(\cdot, q)$ ,  $q \in Q$ . Define  $I(q) = \nu_q(\mathbb{R}^n) \setminus \{q\}$ , that is, the set of discrete states “reachable in one step” from  $q$ . We require that for each  $q \in Q$  and each  $p \in I(q)$  there exist closed sets

$$M_{q,p} \equiv \nu_q^{-1}(p).$$

The sets  $\partial M_{q,p}$  are called the *switching boundaries* of the automaton  $A$ . Define  $M_q = \bigcup_{p \in I(q)} M_{q,p}$  and define the domain of capture of state  $q$  by

$$C(q) \equiv \mathbb{R}^n \setminus M_q = \{x \in \mathbb{R}^n \mid \nu(x, q) = q\}.$$

The equations of motion are

$$\begin{aligned} \dot{x}(t) &= f(x(t), q(t)), \\ q(t) &= \nu(x(t), q(t^-)), \end{aligned}$$

with initial condition  $[x_0, q_0] \in \bigcup_{q \in Q} C(q) \times \{q\}$ . The notation  $t^-$  indicates that the discrete state is piecewise right-continuous. Thus, starting at  $[x_0, i]$ , the continuous state trajectory  $x(\cdot)$  evolves according to  $\dot{x} = f(x, i)$ . If  $x(\cdot)$  hits some  $\partial M_{i,j}$  at time  $t_1$ , then the state becomes  $[x(t_1), j]$ , from which the process continues.

Tavernini places restrictions on the model above. First, for each  $q \in Q$  and  $p \in I(q)$ , the set  $M_{q,p}$  is required to be connected and there must exist a smooth function  $g_{q,p} : \mathbb{R}^n \rightarrow \mathbb{R}$  with 0 in its image a regular value such that

$$M_{q,p} = \{x \in \mathbb{R}^n \mid g_{q,p}(x) \geq 0\}.$$

Thus,  $\nu_q^{-1}(p)$  is an  $n$ -submanifold of  $\mathbb{R}^n$  with boundary

$$\partial M_{q,p} = \{x \in \mathbb{R}^n \mid g_{q,p}(x) = 0\},$$

which is an  $(n - 1)$ -submanifold of  $\mathbb{R}^n$ .

Also, Tavernini places the following three key restrictions on differential automata:

1. Define  $\alpha_q = \min\{\text{dist}(M_{q,p}, M_{q,p'}) \mid p, p' \in I(q), p \neq p'\}$ . We require that

$$\alpha(A) \equiv \min_{q \in Q} \alpha_q > 0$$

be satisfied. That is, the distance between any two sets with different discrete transitions is bounded away from zero.

2. Define  $\beta_{q,p} = \min\{\text{dist}(\partial M_{q,p}, \partial M_{p,p'}) \mid p' \in I(p)\}$ . We require that the inequality

$$\beta(A) \equiv \min_{q \in Q} \min_{p \in I(q)} \beta_{q,p} > 0$$

be satisfied. That is, after a discrete transition, the next set from which another discrete transition takes place is at least a fixed distance away.

3. The assumption on  $\alpha(A)$  is such that  $C(q)$  is an open set with boundary  $\partial C(q) = \partial M_q = \bigcup_{p \in I(q)} \partial M_{q,p}$ . We require that the inclusions

$$\partial M_{q,p} \subset C(p), \quad p \in I(q), q \in Q,$$

be satisfied. That is, after a discrete transition the continuous state is in an open set on which the dynamics are well-defined.

We refer to the above as the TDA model, for Tavernini's differential automata.

In [34], Tavernini uses the above assumptions to prove results about the trajectories of differential automata and the numerical computation of their trajectories.

### 3.2 Back-Guckenheimer-Myers Model

The framework proposed by Back, Guckenheimer, and Myers in [3] is similar in spirit to the TDA model. The model is more general, however, in allowing “jumps” in the continuous state-space and setting of parameters when a switching boundary is hit. This is done through *transition functions* defined on the switching boundaries. Also, the model allows a more general state space.

More specifically, the model consists of a state space

$$S = \bigcup_{q \in Q} S_q, \quad Q \simeq \{1, \dots, N\},$$

where each  $S_q$  is a connected, open set of  $\mathbb{R}^n$ . Notice that the sets  $S_q$  are not required to be disjoint.

The continuous dynamics are given by vector fields  $f_q : S_q \rightarrow \mathbb{R}^n$ . Also, one has open sets  $U_q$  such that  $\bar{U}_q \subset S_q$  and  $\partial U_q$  is piecewise smooth. For  $q \in Q$ , the transition functions

$$G_q : S_q \rightarrow S \times Q$$

govern the jumps that take place when the state in  $S_q$  hits  $\partial U_q$ . They must satisfy  $\pi_1(G_q(x)) \in \bar{U}_{\pi_2(G_q(x))}$ , where  $\pi_k$  is the  $k$ th coordinate projection function. Thus,  $\pi_1(G_q(x))$  is the “continuous part” and  $\pi_2(G_q(x))$  is the “discrete part” of the transition function.

The dynamics are as follows. The state starts at point  $x_0$  in  $U_i$ . It evolves according to  $\dot{x} = f_i(x)$ . If  $x(\cdot)$  hits some  $\partial U_i$  at time  $t_1$ , then the state instantaneously jumps to state  $\xi$  in  $\bar{U}_j$ , where  $G(x(t_1)) = (\xi, j)$ . From there, the process continues.

It is assumed that the *switching boundaries*  $\partial U_q$  have a concrete representation in terms of the zeros of

$$h_q \equiv \min\{h_{q,1}, \dots, h_{q,N_q}\}.$$

where the  $h_{q,i} : S_q \rightarrow \mathbb{R}$  are smooth. The convention then is such that  $h_q > 0$  on  $U_q$ .

We refer to the above model (simplified from the one in [3]) as the BGM model.

The paper [3] presents computer tools that have been developed by its authors for the simulation of such hybrid systems.

### 3.3 Nerode-Kohn Model

In [29], Nerode and Kohn take an automata-theoretic approach to systems composed of interacting ODEs and finite automata (FA). They develop many models from this

approach, but to keep the discussion germane to that so far, we discuss only the so-called “event-driven, autonomous sequential deterministic model” [29, p. 331]. The model consists of three basic parts: plant, digital control automaton, and interface. In turn, the interface is comprised of an analog-to-digital (AD) converter and digital-to-analog (DA) converter. We refer to it as the NKSD (for sequential deterministic) model.

The plant is modeled as Equation (2.2). It is considered to be an input/output automaton as follows. The states of the system are merely the usual plant states, members of  $\mathbb{R}^n$  [29, p. 333]. The input alphabet is formally taken to be the set of members of  $(u(\cdot), \delta_k)$  where  $\delta_k$  is a positive scalar and  $u(\cdot)$  is a member of the set of piecewise right-continuous functions mapping  $\mathbb{R}^+$  to  $U$ . Let  $PU$ , for piecewise  $U$ , denote the latter set. Suppose the plant is in state  $x_k$  at time  $t_k$ . The “next state” of the transition function from this state with input symbol  $(u(\cdot), \delta_k)$  is given by  $x_{k+1} \equiv x(t_k + \delta_k)$ , where  $x(\cdot)$  is the solution on  $[t_k, t_k + \delta_k]$  of

$$\dot{x}(t) = f(x(t), u(t - t_k)), \quad x(t_k) = x_k.$$

Setting  $t_{k+1} = t_k + \delta_k$ , the process is continued.

The digital control automaton is a quintuple  $(Q, I, O, \nu, \eta)$ , consisting of the state space, input alphabet, output alphabet, transition function, and output function, respectively. In general,  $Q$ ,  $I$ , and  $O$  may be arbitrary subsets of  $\mathbb{Z}^+$ . However, the interesting case is when these sets are finite and the equations represent a FA with output, which is discussed below. In any case, the functions involved are  $\nu : Q \times I \rightarrow Q$  and  $\eta : Q \times I \rightarrow O$ . The FA may be thought of as operating in “continuous time” by the convention that the state, input, and output symbols are piecewise right-continuous functions:

$$\begin{aligned} q(t) &= \nu(q(t^-), i(t)), \\ o(t) &= \eta(q(t), i(t)). \end{aligned}$$

Here, the state  $q(t)$  changes only when the input symbol  $i(t)$  changes.

It remains to couple these two “automata.” This is done through the interface by introducing maps  $AD : Y \times Q \rightarrow I$  and  $DA : O \rightarrow PU$ . The  $AD$  symbols are determined by (FA-state-dependent) partitions of the output space  $Y$ . These partitions are not allowed to be arbitrary, but are the “essential parts” of small topologies placed on  $Y$  for each  $q \in Q$ . We explain this later. To each  $o \in O$  is associated an open set of  $PU$ . The  $DA$  signal corresponding to output symbol  $o$  is chosen from this open set of plant inputs. The scalar  $\delta_k$  is a formal construct, denoting the time until the next “event.”

Briefly, the combined dynamics is as follows. Assume the continuous state is evolving according to Equation (2.2) and that the FA is in state  $q$ . Then  $AD(\cdot, q)$  assigns to output  $y(t)$  a symbol from the input alphabet of the FA. When this symbol changes, the FA makes the associated state transition, causing a corresponding change in its output symbol  $o$ . Associated with this symbol is a control input,  $DA(o)$ , which is applied as input to the differential equation until the input symbol of the FA again changes.

Now, we explain what is meant by the “small topologies” mentioned above, concentrating on the  $AD$  map. Nerode and Kohn introduce topologies that make each mapping  $AD_q \equiv AD(\cdot, q)$ ,  $q \in Q$ , continuous as follows:

1. First, take any finite open cover of the output space:  $Y = \bigcup_{i=1}^d \mathcal{A}_i$ , where the  $\mathcal{A}_i$  are open in the given topology of  $Y$ .
2. Next, find the so-called *small topology*,  $\mathcal{T}_Y$ , generated by the subbasis  $\mathcal{A}_i$ . This topology is finite and its open sets can be enumerated, say, as  $\mathcal{B}_1, \dots, \mathcal{B}_K$ .
3. Next, find all the non-empty *join irreducibles* in the collection of the  $\mathcal{B}_i$  (that is, all non-empty sets  $\mathcal{B}_j$  such that if  $\mathcal{B}_j = \mathcal{B}_k \cup \mathcal{B}_l$ , then either  $\mathcal{B}_j = \mathcal{B}_k$  or  $\mathcal{B}_j = \mathcal{B}_l$ ). There are a finite number of such join irreducibles, denoted  $\mathcal{C}_1, \dots, \mathcal{C}_M$ .
4. Without loss of generality, let the set of symbols be  $I = \{1, \dots, M\}$  and define the  $AD_q(y) = i$  if  $\mathcal{C}_i$  is the smallest open set containing  $y$ .
5. Create a topology,  $\mathcal{T}_I$ , on  $I$  as follows. For each  $i \in I$ , declare  $\mathcal{D}_i = \{j \mid \mathcal{C}_j \subset \mathcal{C}_i\}$  to be open. Let  $\mathcal{T}_I$  be the topology generated by the  $\mathcal{D}_i$ .

The sets  $AD_q^{-1}(i)$ ,  $i \in I$  are the essential parts mentioned above. For a verification that  $AD_q$  is continuous, as well as other results on  $AD$  and  $DA$  maps, see [7].

The Nerode-Kohn paper develops the underpinning of a theoretical framework for the hybrid continuous/rule-based controllers used by Kohn in applications. Continuity in the small topologies associated with the  $AD$  and  $DA$  maps above plays a vital role in the theory of those controllers. See [29] and the references therein for details.

### 3.4 Brockett's Model

Several models of hybrid systems are described in [12]. We only discuss those that combine ODEs and discrete phenomena since that is our focus here.

Brockett introduces a “type  $D$  hybrid system” as follows:

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t), z[p]), \\ \dot{p}(t) &= r(x(t), u(t), z[p]), \\ z[p] &= \nu(x[t_p], z[p], v[p]), \end{aligned}$$

where  $x(t) \in X \subset \mathbb{R}^n$ ,  $u(t) \in U \subset \mathbb{R}^m$ ,  $p(t) \in \mathbb{R}$ ,  $v[p] \in V$ ,  $z[p] \in Z$ ,  $f : \mathbb{R}^n \times \mathbb{R}^m \times Z \rightarrow \mathbb{R}^n$ ,  $r : \mathbb{R}^n \times \mathbb{R}^m \times Z \rightarrow \mathbb{R}$ , and  $\nu : \mathbb{R}^n \times Z \times V \rightarrow Z$ . Here,  $X$  and  $U$  are open subsets of  $\mathbb{R}^n$  and  $\mathbb{R}^m$ , respectively. In general,  $V$  and  $Z$  may be arbitrary subsets of  $\mathbb{Z}^+$ , but we deal with the case where they are finite. The notation  $[p]$  denotes the greatest integer less than or equal to  $p$  and  $[t_p]$  denotes the value of  $t$  at which  $p$  most recently became an integer;  $z[p]$ , e.g., is short for  $z([p(t)])$ . The “rate equation”  $r$  is required to be nonnegative for all arguments, but need have no upper bound imposed on it. We denote such a system as BD, for Brockett's type  $D$  model.

The first equation represents the continuous dynamics and the last equation the “symbolic processing” done by the system. The input  $u(t)$  is the continuous control exercised at time  $t$ ; the input  $v[p]$  is the  $p$ th symbolic or discrete control, which is exercised at the times when  $p$  passes through integer values. Thus, Brockett has combined continuous and discrete dynamics by the inclusion of the special “timer” variable  $p$ , the two interacting whenever  $p$  takes on an integer value.

In general, one may also introduce continuous and symbolic output maps

$$\begin{aligned} y(t) &= c(x(t), z[p]), \\ o[p] &= \eta(y[t_p], z[p]). \end{aligned}$$

In this case, one may limit  $f$  or  $r$  by allowing them to depend only on  $y$  instead of the full state  $x$ .

Brockett also has a simpler “type  $B$  hybrid system” in which the  $\nu$  equation does not appear and the symbolic control  $v[p]$  replaces  $z[p]$  in the first two equations. Finally, he generalizes BD to the case of “hybrid system with vector triggering” in which one replaces the single rate and symbolic equations with a finite number of such equations.

In [12], Brockett gives many examples of devices modeled with these systems of equations, including buffers, stepper motors, and transmissions.

### 3.5 Discussion

In the sequel, we explore the capabilities of the four hybrid systems models, TDA, BGM, NKSD, and BD, described above. Clearly, these models were developed for different purposes with assumptions arising accordingly. Nevertheless—and for expediency—we note some containment relations among these models.

Here,  $A$  *contains*  $B$  means that every system described by the equations of model  $B$  can be described by the equations of model  $A$ . When the equations of a model describe a system, we say that the model *implements* that system.

First, since we are not interested in control in this paper, we develop autonomous versions of the models NKSD and BD above, in which the control inputs are replaced by fixed functions of state. (The reader interested in control of hybrid systems should consult [9, 12, 16].)

For instance, here is an autonomous version of NKSD, which we refer to as NKAUT:

$$\begin{aligned}\dot{x}(t) &= f(x(t), q(t)), \\ q(t) &= \nu(q(t^-), AD(x(t), q(t^-))),\end{aligned}$$

where  $x(t) \in \mathbb{R}^n$ ,  $q(t) \in Q \simeq \{1, \dots, N\}$ . Here,  $f : \mathbb{R}^n \times Q \rightarrow \mathbb{R}^n$ ,  $\nu : Q \times I \rightarrow Q$ , and  $AD : \mathbb{R}^n \times Q \rightarrow I \simeq \{1, \dots, M\}$ . Note that we have incorporated the output equations into the  $f$ ,  $\nu$ , and  $AD$  functions. The  $AD$  map is restricted as discussed in Section 3.3.

Here is an autonomous version of the BD model, which we refer to as BAUT:

$$\begin{aligned}\dot{x}(t) &= f(x(t), z[p]), \\ \dot{p}(t) &= r(x(t), z[p]), \\ z[p] &= \nu(x[t_p], z[p], [p]),\end{aligned}$$

where  $x(t) \in \mathbb{R}^n$ ,  $p(t) \in \mathbb{R}$ ,  $z[p] \in Z \simeq \{1, \dots, N\}$ ,  $f : \mathbb{R}^n \times Z \rightarrow \mathbb{R}^n$ ,  $r : \mathbb{R}^n \times Z \rightarrow \mathbb{R}$ , and  $\nu : \mathbb{R}^n \times Z \times \mathbb{Z} \rightarrow Z$ . As in BD,  $r$  is restricted to be nonnegative.

By construction, NKSD contains NKAUT and BD contains BAUT. Note also that BAUT is distinct from the TDA and NKAUT models since, for instance, it allows arbitrary dependence of the discrete dynamics  $\nu$  on  $x[t_p]$ , which can lead to partitions not permitted by the other two models. We have other containment relations as follows.

**Remark** *BGM contains TDA.*

**Proof** Given an arbitrary TDA equation, choose, in the BGM model,  $S_q = \mathbb{R}^n$ ,  $U_q = C(q) = \mathbb{R}^n \setminus M_q$ ,  $f_q = f(\cdot, q)$ ,  $G_q(x) = (x, p)$  if  $x \in M_{q,p}$ , and  $h_{q,p} \equiv -g_{q,p}$  for all  $q \in Q$ ,  $p \in I(q)$ .  $\square$

**Remark** *NKAUT contains TDA.*

**Proof** Suppose we are given an arbitrary TDA equation (i.e., a differential automaton  $A$ ). Let primed symbols denote those in the NKAUT model with the same notation as those for the differential automaton. Set  $Q' = Q$ ,  $f'(\cdot, q) = f(\cdot, q)$ ,  $q \in Q'$ . This duplicates the continuous dynamics.

Now, for each  $q \in Q'$ , choose the small topology on  $\mathbb{R}^n$

$$\mathcal{T}_q = C(q) \cup \bigcup_{p \in I(q)} M_{q,p}^\epsilon,$$

where  $0 < \epsilon < \alpha(A)/3$  and

$$M_{q,p}^\epsilon \equiv \{x \in \mathbb{R}^n \mid \text{dist}(x, M_{q,p}) < \epsilon\}.$$

The non-empty join irreducibles are  $C(q)$  and  $M_{q,p}^\epsilon$ ,  $A_{q,p}^\epsilon$ ,  $p \in I(q)$ , where

$$A_{q,p}^\epsilon \equiv M_{q,p}^\epsilon \setminus M_{q,p}.$$

Let  $i_{q,p}$ ,  $j_{q,p}$ ,  $k_q$  denote the symbols associated with the join irreducibles  $M_{q,p}^\epsilon$ ,  $A_{q,p}^\epsilon$ , and  $C(q)$ , respectively. Defining

$$\begin{aligned} \nu'(q, i_{q,p}) &= p, \\ \nu'(q, j_{q,p}) &= q, \\ \nu'(q, k_q) &= q, \end{aligned}$$

duplicates the discrete dynamics.  $\square$

**Remark** *BGM contains BAUT.*

**Proof** Given an arbitrary BAUT equation, choose, in the BGM model,

$$\begin{aligned} S_1 &= \mathbb{R}^n \times \mathbb{R} \times \mathbb{R} \times \mathbb{R}, \\ U_1 &= \mathbb{R}^n \times (-\infty, 1) \times \mathbb{R} \times \mathbb{R}, \\ f_1(x, q, p, z) &= [f(x, z), r(x, z), 0, 0], \\ G_1(x, q, p, z) &= (x, 0, p + 1, \nu(x, z, p), 1), \\ h_1(x, q, p, z) &= 1 - q. \end{aligned}$$

$\square$

Notice the last proof shows that in the BGM model, setting parameters on hitting switching boundaries can be implemented with the transition functions. Note also that unlike the first two proofs, the last construction uses a different (but equivalent) state space for BGM and BAUT. In any case, we do not use the fact that BGM contains BAUT in further results. Also, we do not compare among BGM, NKSD, and BD here.

Summarizing results needed later, the BGM and NKSD models contain the TDA model; BD contains BAUT. In the sequel, then, the presentation concentrates on the TDA and BAUT models since all capabilities possessed by these models will automatically be possessed by the four hybrid systems models reviewed above. Extra capabilities of the BGM model are noted as warrants.

## 4 Notions of Simulation

In dynamical systems, simulation is captured by the notions of topological equivalence and homomorphism [31, 17, 15]. One can extend these notions to systems with inputs and outputs by also allowing memoryless, continuous encoding of inputs, outputs, and initial conditions.

In computer science, simulation is based on the notion of “machines that perform the same computation.” This can be made more precise, but is not reviewed here [5, 24].

Other notions of simulation (for discrete dynamical systems) appear in [20]. All these notions, however, are “homogeneous,” comparing continuous systems with continuous ones or discrete with discrete. One that encompasses simulation of a discrete dynamical system by a continuous dynamical system is required here.

One notion that associates discrete and continuous dynamical systems is global section [31]. The set  $S_X \subset X$  is a *global section* of the continuous dynamical system  $[X, \mathbb{R}^+, f]$  if there exists a  $t_0 \in \mathbb{R}^+$  such that

$$S_X = \{f(P, kt_0) \mid k \in \mathbb{Z}^+\},$$

where  $P$  is a set containing precisely one point from each of the trajectories  $f(p, \mathbb{R}^+)$ ,  $p \in X$ . Using this for guidance, we define

**Definition (S-simulation)** *A continuous dynamical system  $[X, \mathbb{R}^+, f]$  simulates via section or S-simulates a discrete dynamical system  $[Y, \mathbb{Z}^+, F]$  if there exist a continuous surjective partial function  $\psi : X \rightarrow Y$  and  $t_0 \in \mathbb{R}^+$  such that for all  $x \in \psi^{-1}(Y)$  and all  $k \in \mathbb{Z}^+$*

$$\psi(f(x, kt_0)) = F(\psi(x), k).$$

Note that surjectivity implies that for each  $y \in Y$  there exists  $x \in \psi^{-1}(y)$  such that the equation holds. Here, continuous partial function means the map from  $\psi^{-1}(Y)$  (as a subspace of  $X$ ) to  $Y$  is continuous.

Intuitively, the set  $V \equiv \psi^{-1}(Y)$  may be thought of as the set of “valid” states; the set  $X \setminus V$  as the “don’t care” states. In dynamical systems,  $V$  may be a Poincaré section;  $X \setminus V$  the set of points for which the corresponding Poincaré map is not defined [17, 18]. In computer science and electrical engineering,  $V$  may be the set of circuit voltages corresponding to a logical 0 or 1;  $X \setminus V$  the voltages for which the logical output is not defined.

S-simulation is a strong notion of simulation. For instance, compare it with topological equivalence. Typically, though, the homogeneous notions of simulation do not expect time to be parameterized the same (up to a constant) for both systems. For example, a universal Turing machine,  $U$ , may take several steps to simulate a single step of any given Turing machine,  $M$ . Moreover, the number of such  $U$  steps to simulate an  $M$  step may change from  $M$  step to  $M$  step. Some of the notions of simulation defined in [20] also allow this generality. Further, the definition of topological equivalence of vector fields (different than for dynamical systems, see [17]) is such that parameterization of time need not be preserved. Thus, following the definitions in [20] one formulates

**Definition (P-simulation)** *A continuous dynamical system  $[X, \mathbb{R}^+, f]$  simulates via points or P-simulates a discrete dynamical system  $[Y, \mathbb{Z}^+, F]$  if there exists a continuous surjective partial function  $\psi : X \rightarrow Y$  such that for all  $x \in \psi^{-1}(Y)$  there is a*

sequence of times  $0 = t_0 < t_1 < t_2 < \dots$ ,  $\lim_{k \rightarrow \infty} t_k = \infty$ , such that

$$\psi(f(x, t_k)) = F(\psi(x), k).$$

One readily checks that S-simulation implies P-simulation. This is a weak notion. For instance, consider the case where  $Y$  is finite,  $|Y| = N$ . Suppose  $[X, \mathbb{R}^+, f]$  has a point  $p$  such that  $|f(p, \mathbb{R}^+)| \geq N$  and  $p = f(p, t_0)$  for some  $t_0 > 0$ . That is, the orbit at point  $p$  is periodic and contains more than  $N$  points. Clearly, one may associate  $N$  distinct points in  $f(p, \mathbb{R}^+)$  with the points in  $Y$ , so that  $[X, \mathbb{R}^+, f]$  P-simulates  $[Y, \mathbb{Z}^+, F]$ . This weakness persists even if  $Y$  is infinite. For example, the simple harmonic oscillator defined on the unit circle,  $X = S^1$ :

$$\begin{aligned} \dot{x}_1 &= x_2, \\ \dot{x}_2 &= -x_1, \end{aligned}$$

along with  $\psi(x) = x_1$  P-simulates every  $[[ -1, 1], \mathbb{Z}^+, F]$ . These arguments also show the weakness of some of the definitions in [20]. Finally, this same example shows P-simulation does not imply S-simulation: the harmonic oscillator above cannot S-simulate any  $[[ -1, 1], \mathbb{Z}^+, F]$  for which 0 is a fixed point and 1 is not a fixed point.

Thus, P-simulation need not correspond to an intuitive notion of simulation. The reason is that one wants, roughly, homeomorphisms from orbits to orbits, not from points to points. As mentioned in Section 2, this is achieved with continuous dynamical systems. However, this is not possible with nontrivial nonhomogeneous systems since a discrete orbit with more than one point is a (countable) disconnected set and any non-constant continuous orbit is an (uncountable) connected set. Thus, there exist homeomorphisms between discrete and continuous orbits only when both are constant.

If  $X$  is connected and  $Y$  is a discrete topological space, this situation exists even with points, i.e., the only continuous functions from  $X$  to  $Y$  are constant functions [27]. One way to remedy this is simply to place topologies on  $X$  and  $Y$  other than their usual topologies, so that continuous maps are possible (cf. Section 3.3). There are several ways to accomplish this. One approach is to use so-called small topologies on  $X$ . Another is to append a single element  $\{\perp\}$  to  $Y$ , which stands for “don’t care” or “continue,” and topologize  $Y' = Y \cup \{\perp\}$ . For more information and other approaches see [7, 29].

Here—and with a view towards simulating systems defined on discrete topological spaces—we strengthen the definition of P-simulation in two ways. First, we require that the “simulated state” be valid on some neighborhood and for at least some minimal time period. Physically, this allows one to use “imprecise sampling” to obtain discrete data, providing a robustness that is lacking in the definition of P-simulation. Second, we require that the “readout times” are exactly those for which  $x(t) \in \psi^{-1}(Y)$ .

**Definition (I-simulation)** *A continuous dynamical system  $[X, \mathbb{R}^+, f]$  simulates via intervals or I-simulates a discrete dynamical system  $[Y, \mathbb{Z}^+, F]$  if there exist a continuous surjective partial function  $\psi : X \rightarrow Y$  and  $\epsilon > 0$  such that  $V \equiv \psi^{-1}(Y)$  is open and for all  $x \in V$  the set  $T = \{t \in \mathbb{R}^+ \mid f(x, t) \in V\}$  is a union of intervals  $(\tau_k, \tau'_k)$ ,  $0 = \tau_0 < \tau'_0 < \tau_1 < \tau'_1 < \dots$ ,  $|\tau'_k - \tau_k| \geq \epsilon$ , with*

$$\psi(f(x, t_k)) = F(\psi(x), k),$$

for all  $t_k \in (\tau_k, \tau'_k)$ .

Clearly I-simulation implies P-simulation. S-simulation and I-simulation, however, are independent notions.

The extra requirement that  $\psi^{-1}(Y)$  be open implies that the inverse images of open sets in  $Y$  are open in  $X$  (and not just in  $\psi^{-1}(Y)$  as before). This is probably too strong a requirement in the case of a general topological space  $Y$ . However, in the case of  $Y$  a discrete topological space, it has the desirable effect that  $\psi^{-1}(y)$  is open for all  $y \in Y$ .

One might also have required an output map that is zero (or any distinguished output value) on the complement of  $T$  and non-zero otherwise. This amounts to, in the case of a universal Turing machine simulating a machine  $M$ , the existence of a distinguished state meaning “a step of the simulated machine is not yet completed.” Here, it is related to the appending of a symbol  $\{\perp\}$  to  $Y$  as above and extending  $\psi : X \rightarrow Y' = Y \cup \{\perp\}$  by defining  $\psi(x) = \{\perp\}$  if  $x \in X \setminus \psi^{-1}(Y)$  [7, 29, 2]. In this case, the requirements on  $\psi$  may be replaced by requiring  $\psi$  to be continuous from  $X$  to  $Y'$  (in a suitable topology) after extension. Finally, if  $X$  is a metric space one could introduce a “robust” version of I-simulation by requiring the inverse image of  $y \in Y$  to contain a ball with at least some minimum diameter.

Below, “simulation” is a generic term, meaning I-simulation, S-simulation, or both. SI-simulation denotes S-simulation and I-simulation. If a machine is equivalent, or simulates one that is equivalent, to a universal Turing machine, one says it has *the power of universal computation*.

## 5 Simulation with Hybrid Systems and Continuous ODEs

In this section we concentrate on general simulation results and the capabilities of hybrid systems and continuous ODEs.

We first construct low-dimensional discrete dynamical systems in  $\mathbb{Z}^n$  that are equivalent to finite automata (FA), pushdown automata (PDA), and Turing machines (TMs). Later, we give some general results for continuous ODEs in  $\mathbb{R}^{2n+1}$  simulating discrete dynamical systems in  $\mathbb{Z}^n$ . Combining allows us to conclude simulation of arbitrary FA, PDA, and TMs. By simulating a universal TM, one obtains continuous ODEs with the power of universal computation. In the process, we also discuss the simulation and computational capabilities of hybrid systems.

### 5.1 Discrete Dynamical Systems Equivalent to FA, PDA, and TMs

We start by showing that every TM is equivalent to a discrete dynamical system in  $\mathbb{Z}^2$  and then consider systems equivalent to PDA and FA. Later, we refine these results to discrete dynamical systems in  $\mathbb{Z}$  equivalent to TMs, PDA, and FA.

The FA, PDA, and TMs considered here are deterministic. Thus their transition functions naturally give rise to discrete dynamical systems. These are defined on state spaces of input strings and states; input strings, states, and stacks; and states, tape head positions, and tapes, respectively.

Here, the states, input strings, stacks, and tape configurations of automata and Turing machines are taken in the discrete topology;  $\mathbb{Z}^n$  as a topological or normed space is considered as a subspace of  $\mathbb{R}^n$  (in particular, it has the discrete topology).

An *inputless* FA (resp. PDA) is one whose input alphabet is empty, i.e., one whose transition function depends solely on its state (resp. state and top stack symbol). See

[19] for precise definitions of FA, PDA, and TM.

- Proposition 5.1**
1. *Every TM is equivalent to a discrete dynamical system in  $\mathbb{Z}^2$ .*
  2. *There is a discrete dynamical system in  $\mathbb{Z}^2$  with the power of universal computation.*
  3. *Every FA and inputless PDA is equivalent to a discrete dynamical system in  $\mathbb{Z}$ . Every PDA is equivalent to a discrete dynamical system in  $\mathbb{Z}^2$ .*

**Proof**

1. Assume the tape alphabet is  $\Gamma = \{\gamma_0, \gamma_1, \dots, \gamma_{m-2}\}$ ,  $m \geq 2$ , with  $\gamma_0$  the blank symbol; and that the set of states is  $Q = \{q_0, \dots, q_{n-1}\}$ ,  $n \geq 1$ . Define  $p = \max\{m, n\}$ .

As is customary, the one-sided infinite tape is stored in two stacks, with the state stored on the top of the right stack. The coding used is  $p$ -ary. In particular, suppose the TM is in configuration  $C$ , with tape

$$\mathcal{T} = \gamma_{i_1}, \dots, \gamma_{i_{N-1}}, \gamma_{i_N}, \gamma_{i_{N+1}}, \dots,$$

head positioned at cell  $N$ , and internal state  $q_j$ . Encode the configuration  $C$  in the integers

$$T_L = f_1(C) = \sum_{k=0}^{N-1} p^k i_{N-k} + p^N (m-1), \quad T_R = f_2(C) = j + \sum_{k=1}^{\infty} p^k i_{N+k}.$$

The second sum is finite since only finitely many tape cells are non-blank. The integer  $(m-1)$  is an end-of-tape marker. The TM is assumed to halt on moving off the left of the tape, so that  $(m-1, T_R)$  in  $\mathbb{Z}^2$  is a fixed point for all valid  $T_R$ . On all other valid configurations,  $C$ , define transition function  $G$  in  $\mathbb{Z}^2$  by

$$G(f_1(C), f_2(C)) = (f_1(C'), f_2(C')),$$

where  $C'$  is the configuration resulting when the next move function of the TM is applied to configuration  $C$ .

2. Use part 1 with any universal TM.
3. The inputless cases are immediate from part 1. For the cases with input, note that we encode the input string in an integer like the left part of the tape of a TM above, the results following.

□

Note that one can perform the above encodings of TMs, FA, and PDA with  $[0, p]$  replacing  $\mathbb{Z}$ . Merely replace  $p$  by  $p^{-1}$  in the formulas. The important thing added is compactness, and other encodings, e.g., with  $[0, 1]$  replacing  $\mathbb{Z}$ , follow similarly. There is a problem using these encodings since two distinct tapes may have the same encoding, e.g.,  $3, 2, 0^\omega$  and  $3, 1^\omega$ . One can get around this by “separating” each tape encoding by replacing  $p$  with  $2p$  and using  $2i$  for the  $i$ th symbol. Namely, the tape of length  $N$ ,  $\mathcal{T} = \gamma_{i_1}, \dots, \gamma_{i_N}$ , is encoded as  $\sum_{k=0}^N (2p)^{-k} 2i_k$ . Such “Cantor encodings”

were used in [32]. We still do not use such encodings here, however, since later we want to ensure a minimum distance between any two tape encodings.

Finally, a wholly different approach is to use encodings inspired by those in [13]. Suppose we are given an arbitrary TM,  $T$ . Let  $q$ ,  $h$ ,  $l$ , and  $r$  be integer codings of its state, position of its read-write head, the parts of the tape on the left and on the right of its head, respectively. A configuration of  $T$  is encoded in the integer  $2^q 3^h 5^l 7^r$ .

More generally, any discrete dynamical system in  $\mathbb{Z}^n$  is equivalent to one in  $\mathbb{Z}$  by using such encodings, viz., by associating  $(i_1, i_2, \dots, i_n)$  with  $p_1^{i_1} p_2^{i_2} \dots p_n^{i_n}$ , where  $p_i$  is the  $i$ th prime.

We could have used such constructions instead of those in Proposition 5.1. However, we retain them since their transition functions have properties which those arising from the “prime encodings” do not (cf. Section 5.3). In any case, we conclude

**Proposition 5.2** *Every TM, PDA, inputless PDA, FA, and inputless FA is equivalent to a discrete dynamical system in  $\mathbb{Z}$ . There is a discrete dynamical system in  $\mathbb{Z}$  with the power of universal computation.*

It is important to note that one can extend the transition functions in  $\mathbb{Z}^n$  above to functions taking  $\mathbb{R}^n$  to  $\mathbb{R}^n$ . We may extend any function  $f : A \subset \mathbb{Z}^n \rightarrow \mathbb{R}^m$  in such a manner, by first extending arbitrarily to domain  $\mathbb{Z}^n$  and then using linear interpolation. Here is an example, used below:

**Example** *A continuous mod function may be defined as follows:*

$$x \bmod_C m \equiv \begin{cases} (\lfloor x \rfloor \bmod m) + x - \lfloor x \rfloor, & 0 \leq \lfloor x \rfloor \bmod m < m - 1, \\ (m - 1)(\lfloor x \rfloor + 1 - x), & \lfloor x \rfloor \bmod m = m - 1. \end{cases}$$

Later results require extensions that are robust to small input errors. That is, one would like to obtain the integer-valued result on a neighborhood of each integer in the domain. For instance, one may define a continuous nearest integer function,  $[\cdot]_C : \mathbb{R} \rightarrow \mathbb{R}$ , that is robust in this manner as follows:

$$[x]_C \equiv \begin{cases} i, & i - 1/3 < x \leq i + 1/3, \\ 3x - 2i - 1, & i + 1/3 < x \leq i + 2/3. \end{cases}$$

More generally, define  $\Pi : \mathbb{R}^n \rightarrow \mathbb{R}^n$ , by

$$\Pi(x) = [[x_1]_C, \dots, [x_n]_C].$$

Then given any function  $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ , with  $f(\mathbb{Z}^n) \subset \mathbb{Z}^m$ , we can define a “robust version” by using the function  $f \circ \Pi$ .

Thus, given  $[A, \mathbb{Z}^+, F]$ ,  $A \subset \mathbb{Z}^n$ , its transition function may be extended to a continuous function from  $\mathbb{R}^n$  to  $\mathbb{R}^n$  which is constant in a neighborhood of each point in  $A$ . Such a remark is actually a byproduct of a more general result needed below [27, p. 216]:

**Fact** *Any continuous function  $f : A \rightarrow \mathbb{R}^m$ ,  $A$  a closed subset of  $\mathbb{R}^n$ , may be extended to a continuous map  $\tilde{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ .*

Throughout the rest of this section we use continuous extensions as in the fact above, the notation  $\tilde{f}$  always denoting such an extension of  $f$ .

## 5.2 The Power of Exact Clocks

Later, we find that the ability to implement precise timing pulses is a strong system characteristic, enabling one to implement equations with powerful simulation capabilities. To this end, define

**Definition (Exact Clock)** *A function  $S : \mathbb{R}^+ \rightarrow \mathbb{Z}$  is an exact  $m$ -ary clock with pulse-width  $T$  or simply  $(m, T)$ -clock if*

1. *It is piecewise continuous with finite image  $Q = \{0, \dots, m-1\}$ ,  $m \geq 2$ .*
2. *For all  $t \in (kT, (k+1)T)$ ,  $S(t) = i$  if  $k \equiv i \pmod{m}$ .*

All four hybrid systems models, TDA, BGM, NKSD, and BD, can implement  $(m, T)$ -clocks, as the results of Section 3.5 and the following shows.

**Example** 1. *The BAUT model implements  $(m, T)$ -clocks: Choose  $Z = \{0, \dots, m-1\}$  and*

$$\begin{aligned} \dot{p} &= 1/T, & p(0) &= 0, \\ z[p] &= (z[p] + 1) \bmod m, & z[0] &= 0. \end{aligned}$$

*Then  $S(t) = z[p(t)]$  is an  $(m, T)$ -clock.*

2. *The TDA model implements  $(m, T)$ -clocks: Set  $p = m$  if  $m$  even,  $p = m + 1$  if  $m$  odd. Choose state space  $\mathbb{R} \times Q$ ,  $Q = \{0, \dots, p-1\}$ . Define the continuous dynamics as*

$$f(x, q) = c_q(-1)^q,$$

*$q \in Q$ . Set  $c_q = 1$  for all  $q$  if  $m$  is even; set  $c_q = 1$  for  $q \in \{0, \dots, m-2\}$ ,  $c_q = 2$  for  $q \in \{m-1, m\}$ , if  $m$  odd. In each case define the switching manifolds by*

$$\begin{aligned} g_{2k, 2k+1}(x) &= x - T, \\ g_{2k+1, (2k+2) \bmod p}(x) &= -x. \end{aligned}$$

*Setting  $x(0) = 0$ ,  $S(t) = q(t)$  and*

$$S(t) = q(t) - (m-1)[q(t)/3(m-1)]_C$$

*are  $(m, T)$ -clocks when  $m$  is even and odd, respectively.*

As an example of the simulation power one obtains with access to an exact clock, consider the following:

**Theorem 5.3** *Every reversible discrete dynamical system  $F$  defined on a closed subset of  $\mathbb{R}^n$  can be  $S$ -simulated by a system of continuous ODEs in  $\mathbb{R}^{2n}$  with a  $(2, T)$ -clock,  $S$ , as input.*

**Proof**

$$\begin{aligned} \dot{x}(t) &= T^{-1}[\tilde{G}(z) - z](1 - S(t)), \\ \dot{z}(t) &= T^{-1}[x - \tilde{H}(x)]S(t), \end{aligned}$$

where  $\tilde{G}$  and  $\tilde{H}$  are continuous extensions of  $G = F(\cdot, 1)$  and  $H = F(\cdot, -1)$ , respectively. Starting this system at  $t = 0$  with  $x(0) = z(0) = x_0$ ,  $x_0 \in \text{domain } G$ , one sees that  $x(2kT) = z(2kT) = G^k(x_0)$ . Here,  $\psi(x, z) = x$  for  $x = z$ ,  $x \in \text{domain } G$ .  $\square$

This theorem shows that exact clocks allow one to  $S$ -simulate arbitrary reversible discrete dynamical systems on closed subsets of  $\mathbb{R}^n$  with a system of ODEs in  $\mathbb{R}^{2n}$ . The idea of turning on and off separate systems of differential equations is key to the simulation. The effect of the simulation is that on alternating segments of time one “computes” the next state, then copies it, respectively. Then, the process is repeated.

One readily sees that the exact way the continuous extensions in the proof are performed is not important.

As seen above each of the four hybrid systems models can implement  $(2, T)$ -clocks. In particular, they can implement a  $(2, T)$ -clock with just a single ODE. Thus the simulations of the theorem can be performed with continuous state space  $\mathbb{R}^{2n+1}$  in each of these cases. Further, they each require only 2 discrete states.

The generality of Theorem 5.3 allows us to conclude

**Corollary 5.4** *Using  $S$ -simulation, any hybrid systems model that implements continuous ODEs and a  $(2, T)$ -clock has the power of universal computation.*

**Proof** Using constructions as in Proposition 5.1, construct a reversible discrete dynamical system in  $\mathbb{Z}^n$  equivalent to a universal, reversible TM (one whose transition function is invertible) [4, 35]. In turn, simulate it using the theorem.  $\square$

However, we want to explore simulation of non-reversible finite and infinite computational machines with hybrid and continuous dynamical systems. First, we show that the ability to set parameters on clock edges is strong.

**Theorem 5.5** *Every discrete dynamical system  $F$  defined on a closed subset of  $\mathbb{R}^n$  can be  $S$ -simulated by a system of continuous ODEs on  $\mathbb{R}^{2n}$  (resp.  $\mathbb{R}^n$ ) with a  $(2, T)$ -clock,  $S$ , as input and the ability to set parameters on clock edges.*

**Proof** Define  $G \equiv F(\cdot, 1)$ . Both systems are initialized at  $t = 0$  with  $c = x(0) = x_0$ ,  $x_0 \in \text{domain } G$ .

1. Initialize  $z(0) = x_0$ .

$$\begin{aligned}\dot{x}(t) &= T^{-1}[\tilde{G}(z) - z](1 - S(t)) \\ \dot{z}(t) &= T^{-1}[x - c]S(t)\end{aligned}$$

The constant  $c$  is set to  $z$  when  $t = kT$ ,  $k$  odd. One sees that  $x(2kT) = z(2kT) = G^k(x_0)$ . Choose  $\psi(x, z) = x$  for  $x = z$ ,  $x \in \text{domain } G$ .

- 2.

$$\dot{x}(t) = T^{-1}[\tilde{G}(c) - c](1 - S(t))$$

The constant  $c$  is set to  $x$  when  $t = kT$ ,  $k$  even. One sees that  $x(2kT) = G^k(x_0)$ . Choose  $\psi(x) = x$ ,  $x \in \text{domain } G$ .

$\square$

Note that if  $F$  is not reversible, forward trajectories of the above systems of equations may merge. This situation is allowed by our definitions. The simplest example of this is  $[\{0, 1\}, \mathbb{Z}^+, F]$  with  $F(0, 1) = F(1, 1) = 0$ .

**Corollary 5.6** *Any hybrid systems model that implements continuous ODEs, a  $(2, T)$ -clock, and setting parameters on clock edges, can  $S$ -simulate any TM, PDA, or FA; and, using  $S$ -simulation, has the power of universal computation.*

**Proof** Combine the theorem and Proposition 5.1.  $\square$

In particular, the BGM model has this power (by defining the appropriate transition functions on the switching boundaries of the TDA  $(2, T)$ -clock given above).

### 5.3 Simulation Without Exact Clocks

Without an exact clock, one's simulation power is limited. However, one can still simulate discrete dynamical systems defined on arbitrary subsets of  $\mathbb{Z}^n$ . Next, we proceed to explicitly show that all four hybrid systems models can simulate any discrete dynamical system on  $\mathbb{Z}^n$ . Indeed, we show that continuous ODEs can simulate them.

In the previous section we used an exact  $(2, T)$ -clock to precisely switch between two different vector fields in order to simulate discrete dynamical systems in  $\mathbb{R}^n$ . Again, the essential idea behind the simulations in this section is to alternately switch between two different vector fields. However, since we are simulating systems in  $\mathbb{Z}^n$ , using "robust versions" of their transition functions, and choosing well-behaved ODEs, it is not necessary to precisely time these switches using an exact clock. Indeed, we can use continuous functions to switch among vector fields.

It is still convenient to ensure, however, that only one vector field is active (non-zero) at any given time. Thus, we would like

**Definition (Inexact Clock)** *An inexact  $(m, T)$ -clock,  $m \geq 2$ , is a continuous function  $S : \mathbb{R}^+ \rightarrow [0, 1]^m$  such that on each interval  $t \in [kT, (k+1)T]$  with  $k \equiv i \pmod{m}$  the following hold:  $S_{j+1}(t) = 0$ ,  $0 \leq j \leq m-1$ ,  $j \neq i$ ;  $S_{i+1}(t) \equiv 1$  on a sub-interval of length greater than or equal to  $T/2$ .*

It is also reasonable to require that transitions between 0 and 1 take place quickly or that there be some minimum separation between the times when  $S_i > 0$ ,  $S_j > 0$ ,  $i \neq j$ . Below, we need an inexact  $(2, T)$ -clock with the latter property.

What is key is that such inexact clocks do not require discontinuous vector fields, discontinuous functions, or discrete dynamics. They can be implemented as follows.

**Example (Inexact  $(2, T)$ -clock)** *Define  $\dot{\tau}(t) = 1/T$ , initialized at  $\tau(0) = 0$ . Now, define*

$$S_{1,2}(\tau) = h_{\pm}[\sin(\pi\tau)],$$

where

$$h_+(r) = \begin{cases} 0, & r \leq \delta/2, \\ 2r/\delta - 1, & \delta/2 < r \leq \delta, \\ 1, & \delta < r, \end{cases}$$

$h_-(r) = h_+(-r)$ , and  $0 < \delta < \sqrt{2}/2$ .

Thus, one can switch between two different systems of ODEs with (Lipschitz) continuous functions of the state of another (Lipschitz) ODE. This is why  $2n+1$  dimensional ODEs are used below to simulate an  $n$ -dimensional discrete dynamical system.

We also need the following technical definitions:

**Definition (Non-degeneracy, finite gain)** *A function  $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ , is non-degenerate (resp. finite gain) if there exist constants  $\beta \geq 0$ ,  $M \geq 0$ , such that*

$$\|x\| \leq M\|f(x)\| + \beta, \quad (\text{resp. } \|f(x)\| \leq M\|x\| + \beta),$$

for all  $x \in X$ .

Now we are ready for our main simulation result:

**Theorem 5.7** *Every discrete dynamical system  $F$  defined on  $Y \subset \mathbb{Z}^n$*

1. *can be SI-simulated by a system of continuous ODEs in  $\mathbb{R}^{2n+1}$ .*
2. *such that  $F(\cdot, 1)$  is finite gain and non-degenerate can be I-simulated by a system of continuous ODEs in  $\mathbb{R}^{2n+1}$ .*
3. *such that  $Y$  is bounded can be SI-simulated by a system of Lipschitz ODEs in  $\mathbb{R}^{2n+1}$ .*

**Proof** Let  $G \equiv F(\cdot, 1)$  and  $0 < \epsilon < 1/3$ .  $S_{1,2}$  and  $\delta$  are as in the preceding example. For each  $y \in Y$ , define the set

$$H_y = \{(x, z, \tau) \mid \|x - y\|_\infty < \epsilon, \|z - y\|_\infty < \epsilon, \sin(\pi\tau) < \delta/2, \tau \bmod_C 2 < 1/2\},$$

Set  $\psi(x, z, \tau) = \Pi(z) = y$  if  $(x, z, \tau) \in H_y$ . Note that the  $\psi^{-1}(y) = H_y$  are open and disjoint.

Initialize  $x(0), z(0), \tau(0)$  in  $\psi^{-1}(y)$ ,  $y \in Y$ .

1. Choose

$$\begin{aligned} \dot{x} &= -\epsilon^{-2}[x - \tilde{G}(\Pi(z))]^3 S_1(\tau), \\ \dot{z} &= -\epsilon^{-2}[z - \Pi(x)]^3 S_2(\tau), \\ \dot{\tau} &= 1. \end{aligned}$$

It is straightforward to verify  $\Pi(z(2k)) = G^k(y)$ ,  $k \in \mathbb{Z}^+$ , and the interval constraint.

2. Let  $\alpha$  and  $L$  be the finite gain, and  $\beta$  and  $M$  the non-degeneracy constants of  $G$  under norm  $\|\cdot\|_\infty$ . Choose

$$\begin{aligned} \dot{x} &= -2\epsilon^{-1}[x - \tilde{G}(\Pi(z))]S_1(\tau), \\ \dot{z} &= -2\epsilon^{-1}[z - \Pi(x)]S_2(\tau), \\ \dot{\tau} &= 1/[1 + (L+1)\|z\|_\infty + \alpha + (M+1)\|x\|_\infty + \beta]. \end{aligned} \tag{5.1}$$

It is straightforward to verify  $\Pi(z(t)) = G^k(y)$  on an interval about the time  $t_k$  where  $\tau(t_k) = 2k$ ,  $k \in \mathbb{Z}^+$ .

3. Let  $\beta = \max\{\|i - j\|_\infty \mid i, j \in Y\}$ . Choose

$$\begin{aligned} \dot{x} &= -2\beta\epsilon^{-1}[x - \tilde{G}(\Pi(z))]S_1(\tau), \\ \dot{z} &= -2\beta\epsilon^{-1}[z - \Pi(x)]S_2(\tau), \\ \dot{\tau} &= 1. \end{aligned} \tag{5.2}$$

It is straightforward to verify  $\Pi(z(2k)) = G^k(y)$ ,  $k \in \mathbb{Z}^+$ , and the interval constraint. □

Note that non-degeneracy and finite gain of the extension  $\tilde{G}$  need not hold for points not in  $Y$ . Note also that the simulations above are “robust” in the sense that there is a neighborhood of initial conditions leading to the correct simulations. The

import of part 2 of the theorem is that if  $G \equiv F(\cdot, 1)$  is non-degenerate and may be extended to a Lipschitz function, then the ODEs used in the I-simulation are also Lipschitz.

Note also that the theorem continues to hold for any discrete dynamical system defined on  $Y \subset \mathbb{R}^n$  such that there is some minimum separation between any two distinct points of  $Y$ .

The discrete dynamical systems equivalent to TMs given by Proposition 5.1 have transition functions that are both finite gain and non-degenerate. Unfortunately, the transition functions of systems equivalent even to PDA need not be Lipschitz. Consider a PDA which pushes a tape symbol  $\gamma$  on input symbol  $i_1$  and pops  $\gamma$  on input symbol  $i_2$  and test with inputs of the form  $i_1^{n+1}, i_1^n i_2$ . One may check that the “prime encodings” mentioned earlier lead to transition functions that are neither finite gain nor non-degenerate.

Thus, relating the theorem back to simulation of TMs, PDA, and FA, we have many results, the most striking of which are:

**Corollary 5.8** *Every TM, PDA, and FA can be SI-simulated by a system of continuous ODEs in  $\mathbb{R}^3$ .*

*Every FA (resp. inputless FA) can be I-simulated (resp. SI-simulated) by a system of Lipschitz continuous ODEs in  $\mathbb{R}^3$ .*

*Using SI-simulation, there is a system of continuous ODEs in  $\mathbb{R}^3$  with the power of universal computation.*

**Proof** Everything is immediate from the theorem and Propositions 5.1 and 5.2 except that the FA transition function is Lipschitz, which is readily checked.  $\square$

Of course, any hybrid systems model that implements continuous (resp. Lipschitz) ODEs has similar powers. In particular, the four reviewed here do.

Finally, all the simulation results for discrete dynamical systems on  $\mathbb{Z}$  can be extended from continuous to smooth vector fields by using  $C^\infty$  interpolation (with so-called “bump” functions [15]) rather than linear interpolation in extending their transition functions and the functions  $[\cdot]_C$  and  $h_\pm$ , and by replacing  $\|\cdot\|_\infty$  with  $\|\cdot\|_2$  in Equation (5.1).

## 6 Implementing Arbiters

In this section, we contrast the capabilities of hybrid and continuous dynamical systems by using the famous asynchronous arbiter problem [8, 23, 36].

We begin in the first subsection with a discussion of the arbiter problem. Next, we prove that one cannot implement an asynchronous arbiter using a system of Lipschitz ODEs continuous in inputs and outputs, i.e., a system of the form of Equation (2.2) with  $f$  Lipschitz in  $x$ , continuous in  $u$  and  $h$  continuous [18, p. 297]. Finally, we show that all four hybrid systems models can implement arbiters, even when their continuous dynamics is a system of Lipschitz ODEs continuous in inputs and outputs.

### 6.1 The Arbiter Problem

The definition and technical specifications of an (asynchronous) arbiter below are adapted from [36].

An *arbiter* is a device that can be used to decide the winner of two-person races. It is housed in a box with two input buttons, labeled  $B_1$  and  $B_2$ , and two output lines,  $W_1$  and  $W_2$ , that can each be either 0 or 1. For ease of exposition, let the vectors

$$B = (B_1, B_2), \quad W = (W_1, W_2)$$

denote the button states and outputs, respectively. There is also a reset button,  $R$ . Below, the buttons  $B_i$ ,  $R$  are taken to be 1 when they are pressed, 0 when they are unpressed.

After the system has been reset, the output should be  $(1, 0)$  if  $B_1$  is pressed before  $B_2$ ; it should be  $(0, 1)$  if  $B_2$  is pressed before  $B_1$ . Let  $T_i$  denote the time that button  $B_i$  is pressed. Then, the function of the arbiter is to make a binary choice based on the value of the continuous variable  $T_1 - T_2$ . If the difference is negative, the output should be  $(1, 0)$ ; if it is positive, the output should be  $(0, 1)$ . Upon reset, the output is set to  $(0, 0)$ .

Here are the arbiter's technical specifications:

- S1.** Pressing the reset button,  $R$ , causes the output to become  $(0, 0)$ , perhaps after waiting for some specified time, denoted  $T_R$ , where it remains until one or both buttons are pressed.
- S2.** The pressing of either or both buttons  $B_i$  causes, after an interval of at most  $T_d$  units, the output to be either  $(0, 1)$  or  $(1, 0)$ ; the output level persists until the next reset input.
- S3.** If  $B_1$  is pressed  $T_a$  seconds or more before  $B_2$  is pressed, then the output will be  $(1, 0)$ , indicating that  $B_1$  was pressed first. Similarly, if  $B_2$  is pressed  $T_a$  seconds or more before  $B_1$  is pressed, then the output will be  $(0, 1)$ , indicating that  $B_2$  was pressed first.
- S4.** If  $B_1$  and  $B_2$  are pressed within  $T_a$  seconds of each other, then the output is either  $(1, 0)$  or  $(0, 1)$ —one does not care which—after the  $T_d$ -second interval.

The arbiter problem is

**Problem (Asynchronous arbiter problem)** *Build a device that meets the specifications S1–S4.*

## 6.2 You Can't Implement an Arbiter with Lipschitz ODEs

In this section, we show that it is impossible to build a device, described as a system of Lipschitz ODEs continuous in the required inputs and outputs, that implements the arbiter specifications.

First we give a generic system of Lipschitz ODEs with the required properties:

$$\begin{aligned} \dot{x}(t) &= f(x(t), B(t)), \\ W(t) &= h(x(t)), \end{aligned} \tag{6.1}$$

where  $x(t) \in \mathbb{R}^n$ ,  $W(t) \in \mathbb{R}^2$ ,  $B(t) \in \{0, 1\}^2$ , with  $B(\cdot)$  piecewise continuous. Each  $f(\cdot, B)$ ,  $B \in \{0, 1\}^2$ , is Lipschitz. Thus, each vector field  $f(\cdot, B)$  defines a continuous dynamical system  $\phi_{(B_1, B_2)}$ , with  $\phi_{(B_1, B_2)}(x_0, T)$  the solution at time  $T$  of

$\dot{x}(t) = f(x(t), B_1, B_2)$  starting at  $x(0) = x_0$ . Further,  $h : \mathbb{R}^n \rightarrow \mathbb{R}^2$  is continuous. Note that the action of the reset button is unmodeled; it is not necessary to the proof, which assumes it remains unpressed on the interval of interest.

Since  $h$  is continuous, there exists a constant  $\delta_{\sqrt{2}} > 0$  such that

$$\|h(x) - h(x')\|_2 < \sqrt{2} \quad \text{whenever} \quad \|x - x'\| < \delta_{\sqrt{2}}, \quad (6.2)$$

Define  $L_W = \sqrt{2}/\delta_{\sqrt{2}}$ .

Now, we are ready to settle the arbiter problem in this framework:

**Theorem 6.1** *For no choice of the values for  $T_a$  and  $T_d$  is it possible to build a device described by Equation (6.1) that meets the arbiter specifications S1–S4.*

**Proof** The proof is by contradiction, assuming there is a device described by Equation (6.1) which satisfies the specifications.

Assume that the arbiter has been reset, is in state  $x(0) = x_0$  at time  $t = 0$  with  $h(x_0) = (0, 0)$ , and that one of the buttons is pressed at time  $t = 0$ . (This is without loss of generality as the equations are autonomous.) Also, assume that the reset button is not pressed until some time  $T_R \gg T_a + T_d$ .

The behavior of the device from  $t = 0$  to  $t = T_R$  is completely determined by which button was pressed first and at what time the second button is pressed (if ever). Therefore, let  $x_\rho(t)$  denote the solution at time  $t$  of Equation (6.1) starting at time  $t = 0$  at state  $x(0) = x_0$  with fixed parameter  $\rho \equiv T_1 - T_2$ . Thus,  $\rho$  represents the difference between the times when  $B_1$  and  $B_2$  are pressed. If  $B_1$  is pressed but  $B_2$  is never pressed, set  $\rho = -\infty$ . If  $B_2$  is pressed but  $B_1$  is never pressed, set  $\rho = \infty$ .

The arbiter specifications require that for  $T_a + T_d \leq t \leq T_R$ ,

$$h(x_\rho(t)) = \begin{cases} (1, 0), & \rho \leq -T_a, \\ (0, 1), & \rho \geq T_a, \\ (1, 0) \text{ or } (0, 1), & \text{otherwise.} \end{cases}$$

These specifications and Lemma A.1 (in the Appendix) are such that for any  $\delta > 0$ , one can find  $-T_a \leq \sigma < \tau \leq T_a$ , with  $\tau - \sigma < \delta$ , and with one of  $h(x_\sigma(T_a + T_d))$ ,  $h(x_\tau(T_a + T_d))$  equal to  $(1, 0)$  and the other equal to  $(0, 1)$ .

Pick  $\delta < \min\{T_a, T_d, 1/L\}$ , where  $L > 0$  is a finite bound of the maximum of the four Lipschitz constants corresponding to each of the  $f(\cdot, B)$ . Define

$$c = \max\{\|f(x_0, 1, 0)\|, \|f(x_0, 0, 1)\|, \|f(x_0, 1, 1)\|\}.$$

Note  $c > 0$ , for otherwise  $h(x_\sigma(t)) = h(x_\tau(t)) = h(x_0)$  for all  $0 \leq t \leq T_R$ , a contradiction.

For ease of notation, let  $F_t, G_t, H_t$  denote the fundamental solutions  $\phi_{(0,1)}(\cdot, t)$ ,  $\phi_{(1,0)}(\cdot, t)$ , and  $\phi_{(1,1)}(\cdot, t)$ , respectively. Also, let  $X_t = x_\sigma(t)$  and  $Y_t = x_\tau(t)$ . Note that  $X_0 = Y_0 = x_0$ .

The proof splits into three cases:

1.  $0 \leq \sigma < \tau \leq T_a$ .
2.  $-T_a \leq \sigma < \tau \leq 0$ .
3.  $-T_a < \sigma < 0 < \tau < T_a$ .

**Case 1.** In this case,

$$\begin{aligned} X_t &= \begin{cases} F_t(x_0), & 0 \leq t \leq \sigma, \\ H_{t-\sigma}(F_\sigma(x_0)), & \sigma \leq t \leq T_R, \end{cases} \\ Y_t &= \begin{cases} F_t(x_0), & 0 \leq t \leq \tau, \\ H_{t-\tau}(F_\tau(x_0)), & \tau \leq t \leq T_R. \end{cases} \end{aligned}$$

Thus,  $X_\sigma = Y_\sigma$ . Now, by Corollary A.3

$$\|Y_\tau - Y_\sigma\| \leq cL^{-1}(e^{L\tau} - e^{L\sigma}).$$

Thus, Lemma A.4 gives

$$\begin{aligned} \|X_{\sigma+T_d} - Y_{\tau+T_d}\| &\leq cL^{-1}(e^{L(\tau-\sigma)} - 1)e^{L\sigma}e^{LT_d}, \\ &\leq cL^{-1}(e^{L\delta} - 1)e^{L(T_a+T_d)}, \\ &\leq c\delta(e-1)e^{L(T_a+T_d)}, \end{aligned}$$

where the last line follows from  $L\delta < 1$ . But by assumption,

$$\sqrt{2} = \|h(X_{\sigma+T_d}) - h(Y_{\tau+T_d})\|_2,$$

so that Equation (6.2) yields

$$K_1 \equiv \sqrt{2} / [cL_W(e-1)e^{L(T_a+T_d)}] \leq \delta.$$

**Case 2.** The argument is similar to Case 1 and yields the same inequality on  $\delta$ .

**Case 3.** In this case,

$$\begin{aligned} X_t &= \begin{cases} G_t(x_0), & 0 \leq t \leq |\sigma|, \\ H_{t-|\sigma|}(G_{|\sigma|}(x_0)), & |\sigma| \leq t \leq T_R, \end{cases} \\ Y_t &= \begin{cases} F_t(x_0), & 0 \leq t \leq \tau, \\ H_{t-\tau}(F_\tau(x_0)), & \tau \leq t \leq T_R. \end{cases} \end{aligned}$$

Note that  $\max\{|\sigma|, |\tau|\} \leq \delta$ . This and Lemma A.2 give

$$\|X_{|\sigma|} - Y_\tau\| \leq \|X_{|\sigma|} - x_0\| + \|Y_\tau - x_0\| \leq 2cL^{-1}(e^{L\delta} - 1)$$

Thus, Lemma A.4 gives

$$\begin{aligned} \|X_{|\sigma|+T_d} - Y_{\tau+T_d}\| &\leq 2cL^{-1}(e^{L\delta} - 1)e^{LT_d}, \\ &\leq 2ce^{LT_d}(e-1)\delta, \end{aligned}$$

where the last line follows from  $L\delta < 1$ . But by assumption,

$$\sqrt{2} = \|h(X_{|\sigma|+T_d}) - h(Y_{\tau+T_d})\|_2,$$

so that Equation (6.2) yields

$$K_3 \equiv 1 / [\sqrt{2}cL_W(e-1)e^{LT_d}] \leq \delta.$$

Thus, choosing  $\delta < \min\{T_a, T_d, 1/L, K_1, K_3\}$ , would have achieved a contradiction in all three cases.  $\square$

The basic argument used above is that one cannot have a continuous map from a connected space (e.g.,  $\mathbb{R}$  containing  $\rho$ ) to a disconnected space (e.g.,  $\{(1,0), (0,1)\}$ ) [27]. Nevertheless, one must prove that the map given by the device is indeed continuous before one makes such an appeal. Above, we have explicitly demonstrated the continuity of the system of switched differential equations describing the arbiter.

### 6.3 Implementing Arbiters with Hybrid Systems

In this section it is shown that each of the hybrid systems models can implement an arbiter. Given the results of Section 3.5, it is enough to implement one using the BAUT and TDA models. However, the problem is such that we must add inputs and outputs to these models, which is done in an obvious way.

In each case, the continuous dynamics is a system of Lipschitz ODEs continuous in inputs and outputs, the essential “resolving power” coming from the mechanisms implementing the discrete dynamics.

We first implement an arbiter with a hybrid system *à la* Brockett:

**Proposition 6.2** *There exists a system of equations in the BAUT model with inputs and outputs that meets the arbiter specifications S1–S4.*

**Proof** We design for  $T_a = T_d/2 = T_m$ .

$$\begin{aligned} \dot{x} &= [2(4z\lfloor p \rfloor - 1) \max(B_1, B_2)T(x)/T_m] (1 - R) - (2x/T_R) R, \\ \dot{p} &= [2B_1(B_1 - B_2)(1 - z\lfloor p \rfloor)/T_m] (1 - R) + (z\lfloor p \rfloor/T_R) R, \\ z\lfloor p \rfloor &= (z\lfloor p \rfloor + 1) \bmod 2, \\ W &= h(x), \end{aligned}$$

where

$$h(x) = \begin{cases} (0, 1), & x \leq -3, \\ (0, |x| - 2), & -3 \leq x \leq -2, \\ (0, 0), & -2 \leq x \leq 2, \\ (x - 2, 0), & 2 \leq x \leq 3, \\ (1, 0), & 3 \leq x, \end{cases}$$

$$T(x) = \begin{cases} 1, & |x| \leq 4, \\ 5 - |x|, & 4 \leq |x| \leq 5, \\ 0, & 5 \leq |x|. \end{cases}$$

Let’s examine these equations when  $B_2$  is pressed at time  $t = 0$ . Let  $T_1 > 0$  denote the time at which  $B_1$  is pressed. The equations are assumed to be properly reset so that without loss of generality, we assume that  $|x(0)| < 1$  and  $p(0) \in [2k, 2k + 1)$ , for some  $k \in \mathbb{Z}^+$ , and  $z\lfloor p(0) \rfloor = 0$ . Also, we assume that the reset button is inactive ( $R = 0$ ) from  $t = 0$  to  $t = t_R > 2T_m$ . In this case, the two equations are simply (no matter when  $B_1$  is pressed)

$$\begin{aligned} \dot{x} &= -2T(x)/T_m, \\ \dot{p} &= 0, \end{aligned}$$

so that  $x(t) < -3$  and hence  $W(t) = (0, 1)$  for  $t \in [2T_m, t_R]$ .

Now, we look at these equations under the same assumptions, excepting  $B_1$  is pressed at  $t = 0$  and  $B_2$  is pressed at  $t = T_2 > 0$ . Now there are two cases:  $T_2 < t_z$  and  $T_2 \geq t_z$ , where  $t_z = [1 - (p - \lfloor p \rfloor)]T_m/2 \leq T_m/2$  is the time when  $z(\lfloor p(t) \rfloor)$  would first equal 1 if  $B_2$  were not pressed before it. In the second case, by time  $t_z$  the equations are

$$\begin{aligned}\dot{x} &= 6T(x)/T_m, \\ \dot{p} &= 0,\end{aligned}$$

so that  $x(t) > 4$  and hence  $W(t) = (1, 0)$  for  $t \in [t_z + T_m, t_R] \supset [2T_m, t_R]$ . In the first case, the first equation remains

$$\dot{x} = -2T(x)/T_m,$$

so that  $x(t) < -3$  and hence  $W(t) = (0, 1)$  for  $t \in [2T_m, t_R]$ .

The reset behavior is readily verified.  $\square$

Now, we implement an arbiter with TDA:

**Proposition 6.3** *There exists a system of equations in the TDA model with inputs and outputs that meets the arbiter specifications S1–S4.*

**Proof** For convenience, define  $T_m = \min\{T_d, T_a\}$ . Define the continuous dynamics,  $f(x, q, B_1, B_2, R)$ , which depends on states  $x \in \mathbb{R}^2$ ,  $q \in \{1, 2, 3\}$ , and inputs  $B_1, B_2$ , and  $R$ , each in  $\{0, 1\}$ , as follows:

$$\begin{aligned}f(x, 1; B_1, B_2, 0) &= (B_1[B_1 - B_2], B_2), \\ f(x, 2; \cdot, \cdot, 0) &= (0, 0), \\ f(x, 3; \cdot, \cdot, 0) &= (0, 0), \\ f(x, \cdot; \cdot, \cdot, 1) &= -\frac{T_m}{\epsilon T_R}x,\end{aligned}$$

with switching boundaries defined as follows:

$$\begin{aligned}g_{1,2}(x) &= 4\epsilon^2 - \|x - (T_m, 0)\|_2^2, \\ g_{1,3}(x) &= x_2 - T_m, \\ g_{2,1}(x) = g_{3,1}(x) &= \epsilon^2 - \|x\|_2^2,\end{aligned}$$

where  $0 < \epsilon < T_m/4$ . Finally, define the output  $W = h(x)$  where

$$h(x) = \begin{cases} (1, 0), & x_2 \leq T_m/2, \\ (1 - 4(x_2/T_m - 1/2), 4(x_2/T_m - 1/2)), & T_m/2 < x_2 < 3T_m/4, \\ (0, 1), & 3T_m/4 \leq x_2. \end{cases}$$

One readily verifies that it behaves correctly.  $\square$

## 7 Conclusions

We explored the simulation and computational capabilities of hybrid systems, which combine continuous dynamics (modeled by ODEs) and discrete dynamics (modeled by finite automata). We concentrated on four hybrid systems models from the control and

dynamical systems literature [34, 3, 29, 12]. The four hybrid systems models, denoted TDA, BGM, NKSD, and BD, were reviewed, compared, and examined throughout. Since we were not interested in control in this paper, autonomous versions of the NKSD and BD models were developed. (The reader interested in control of hybrid systems should consult [9, 12, 16].)

We defined notions of simulation of a discrete dynamical system by a continuous dynamical system. S-simulation, or simulation via section, was motivated by the definition of global section in dynamical systems [31]. Relaxing this to allow different parameterizations of time we considered P-simulation, which was seen to be weak. To remedy this, we defined I-simulation, or simulation via intervals. Both S-simulation and I-simulation imply P-simulation. S-simulation and I-simulation are independent notions.

We then showed that hybrid systems models with the ability to implement an exact clock can simulate fairly general discrete dynamical systems. Namely, we demonstrated that such systems can S-simulate arbitrary reversible discrete dynamical systems defined on closed subsets of  $\mathbb{R}^n$ . These simulations require ODEs in  $\mathbb{R}^{2n}$  with the exact clock as input. Each of the four hybrid systems models can implement exact clocks.

Later, we found that one can simulate arbitrary discrete dynamical systems defined on subsets of  $\mathbb{Z}^n$  without the capability of implementing an exact clock. Instead, one can use an approximation to an exact clock, implemented with a one-dimensional Lipschitz ODE. The result is that we can perform SI-simulations (resp. I-simulations) using continuous (resp. Lipschitz) ODEs in  $\mathbb{R}^{2n+1}$ .

Turning to computational abilities, we saw that there are systems of continuous ODEs possessing the ability to SI-simulate arbitrary pushdown automata and Turing machines. Finite automata may be SI-simulated with continuous, Lipschitz ODEs. By SI-simulating a universal Turing machine, we concluded that there are ODEs in  $\mathbb{R}^3$  with continuous vector fields possessing the power of universal computation. Further, the ODEs simulating these machines may be taken smooth and do not require the machines to be reversible (cf. [26, p. 228]).

The import of S-simulation here is that such simulations take only “linear time” [13]. The import of I-simulation is that the readout times for which the state/tape is valid are non-empty intervals. Indeed, the intervals are at least some minimum length. Also, the simulations were “robust” in the sense that they can tolerate small errors in the coding of the initial conditions. Though not required by our definitions, these contained balls of at least some minimum diameter.

Finally, we showed that hybrid systems are strictly more powerful than Lipschitz ODEs in the types of systems they can implement. For this, we used a nontrivial example: the famous asynchronous arbiter problem. First, we settled the problem in an ODE framework by showing one cannot build an arbiter with devices modeled by Lipschitz ODEs continuous in inputs and outputs. Then, we showed that each of the four models of hybrid systems can implement arbiters, even when their continuous dynamics are modeled by Lipschitz ODEs continuous in inputs and outputs.

We now turn to some discussion. Our simulation of arbitrary Turing machines was announced in [6]. It is now a special case of the current results. These results imply that, in general, questions regarding the dynamical behavior of hybrid systems with continuous ODEs—and even well-behaved ODEs themselves—are computationally undecidable. See [25, 26] for a discussion of such questions.

The explicit formulation and solution of the asynchronous arbiter problem in an ODE framework appears to be new. It is excerpted from [8], which also discusses bounds on the performance of systems approximating arbiter behavior, arising from the explicit proof. Specifically, while the proof prohibits the construction of an arbiter with  $T_d = O(1)$ , it does not prohibit an arbitration device with  $T_d = O(\ln(1/\rho))$ . Such a device is given in [8]. Finally, note that in our ODE model, the inputs  $B_i$  were assumed to be ideal in the sense that they switch from 0 to 1 instantaneously. Imposing continuity assumptions on  $B$  as signals in  $[0, 1]^2$  leads to a similar result.

To demonstrate the computational capabilities of hybrid and continuous dynamical systems summarized above, we constructed low-dimensional discrete dynamical systems in  $\mathbb{Z}^n$  equivalent to Turing machines (TMs), pushdown automata (PDA), and finite automata (FA). It is well-known that certain discrete dynamical systems are equivalent to TMs and possess the power of universal computation (see, e.g., [25, 32, 13]). Our systems were constructed with the goal of simulation by continuous/Lipschitz ODEs in mind. One notes that while it is perhaps a trivial observation that there are systems of (Lipschitz) ODEs with the power of universal computation—just write down the ODEs modeling your personal computer—this requires a system of ODEs with a potentially infinite number of states.

The best definition of “simulation” is not apparent. While stated in terms of our definitions of simulation, the simulation results of Section 5 are intuitive and would probably continue to hold under alternate definitions of simulation.

Related to our general simulation results is a theorem by N. P. Zhidkov [37] (see also [31, p. 135]), that states if a reversible discrete dynamical system is defined on a compact subset  $K \subset \mathbb{R}^n$ , then there exists on a subset of  $\mathbb{R}^{2n+1}$  a reversible continuous dynamical system that is defined by ODEs and has  $K$  as a global section.

It is possible to take a different approach than the one in Section 5 and construct smooth systems of ODEs with inputs that “simulate” finite automata. For instance, in [10] Brockett used a system of his so-called double-bracket equations (also see [11]) to “simulate” the step-by-step behavior of a FA. This was done by coding the input symbols of the FA in a function of time that is the “control input” to a system of double-bracket equations. Specifically, if the input alphabet is  $I = \{u_1, \dots, u_m\}$ , the input string  $u_{i_0}, u_{i_1}, u_{i_2}, \dots$  is encoded in a time function,  $u(t)$ , that is  $i_k$  on the intervals  $[2kT, (2k+1)T]$  and zero otherwise. In this paper, we encoded the full input string in the initial condition of our simulations.

In [10], Brockett was interested in the capabilities of his double-bracket equations. However, the resulting “simulations” of FA happen to behave poorly with respect to our definitions of simulation. Nevertheless, the key idea of his simulations of FA is that the input coding,  $u(t)$ , is used in such a way that it alternately switches between two different systems of double-bracket equations. This idea is critical in our simulations of discrete dynamical systems with ODEs.

It is not hard to see that one could use the same approach as that in [10] but more well-behaved systems of ODEs to simulate the step-by-step behavior of FA. Consider a FA with transition function  $\delta$ , states  $Q = \{q_1, \dots, q_n\}$ , and input alphabet  $I$  as above. Code state  $q_i$  as  $i$  and consider the first two equations of Equation (5.2). Choose  $\beta = n$  and replace, respectively,  $S_1$ ,  $S_2$ , and  $G$  with  $h_+(u(t))$ ,  $h_-(u(t) - 1)$ , and

$$D : \{1, \dots, n\} \times \{1, \dots, m\} \rightarrow \{1, \dots, n\},$$

defined by  $D(i, j) = k$  if  $\delta(q_i, u_j) = q_k$ . The result is that any FA may be SI-simulated

by a system of ODEs in  $\mathbb{R}^2$  with input. This was also announced in [6].

In [2], it is shown that so-called piecewise-constant derivative systems (PCDs) in  $\mathbb{R}^3$  can “simulate” arbitrary inputless FA, inputless PDA, and TMs. Briefly, the notion of simulation used is that of I-simulation excepting as follows. First, the intervals in  $T$  can be open, closed, or half-closed; ‘ $\leq$ ’ may replace ‘ $<$ ’ in the constraints on  $\tau_k$ ,  $\tau'_k$ ; and there is no  $\epsilon$  constraint. Also, there is no continuity constraint on  $\psi$  and for each  $y \in Y$  there need exist only one point in  $\psi^{-1}(y)$  for which the equation holds. However, there is the constraint that each  $\psi^{-1}(y)$  is convex and relatively-open (i.e., open in the subspace of its affine hull). For convenience, we refer to this notion as AM-simulation. Since our I-simulations in Theorem 5.7 had  $\psi^{-1}(y)$  open and convex, they are AM-simulations (here we are thinking of  $\tau$  in  $\mathbb{R}/2\mathbb{Z}$ , or in a circle embedded in  $\mathbb{R}^2$ , with appropriate changes).

Convexity of  $\psi^{-1}(y)$  may be a desirable property. For instance, it excludes simulation of FA by “unraveling” their transition diagrams into trees, a simple example of which is recounted in [2]. On the other hand, consider the case of a universal TM,  $U$ , simulating an inputless FA,  $A$ . Certainly, there could be many distinct configurations of  $U$  in which the current state of  $A$  is written on, say, its first tape cell. Then, even if the inverse images of the configurations of  $U$  are convex, the inverse images of the valid configurations with, say,  $q$  in the first tape cell need not be, preventing indirect AM-simulation of  $A$  through AM-simulation of  $U$ . In any case, we could have added the constraint that each  $\psi^{-1}(y)$  be convex to our definitions of simulation with little change in any of our results.

Finally, in [2] Asarin and Maler use three-dimensional PCDs to AM-simulate inputless FA. They also point out that three dimensions are necessary in order to AM-simulate, with autonomous ODEs, inputless FA whose transition graphs are not planar. While their argument is fine, the transition graphs of deterministic inputless FA are always planar and it is straightforward to construct PCDs (and continuous ODEs) in two dimensions that AM-simulate such FA. Moreover, even though the transition graphs of FA (with inputs) need not be planar, their argument does not contradict the result in  $\mathbb{R}^2$  derived in this section, since it uses non-autonomous ODEs.

## Acknowledgements

This work was supported by the Army Research Office and the Center for Intelligent Control Systems under contracts DAAL03-92-G-0164 and DAAL03-92-G-0115. Foremost, thanks to Sanjoy K. Mitter for his guidance and support. Thanks also to John Wyatt, who presented the arbiter problem and the challenge to produce an ODE-based model and proof in his nonlinear systems course at MIT. The author would also like to thank Eduardo Sontag for an engaging discussion on analog computation and Sandro Zampieri for one on definitions of simulation. Also, thanks are due to Mitch Livstone, Ted Theodosopoulos, and the anonymous reviewers, whose suggestions were instrumental in improving the final version of the paper. Finally, thanks to Anil Nerode for some last-minute advice.

## Appendix

**Lemma A.1** *If  $X$  is a connected metric space,  $Y$  is a discrete topological space with two points, and  $f : X \rightarrow Y$  is surjective, then for every  $\delta > 0$  one can find  $x, z \in X$  such that  $d(x, z) < \delta$  and  $f(x) \neq f(z)$ .*

**Proof** Assume the contrary. Then for all  $x \in X$ ,  $f(B_\delta(x)) = \{f(x)\} \subset V$ , where  $B_\delta(x)$  denotes the ball of radius  $\delta$  about  $x$  and  $V$  is any open set about  $f(x)$  in  $Y$ . Thus,  $f$  is continuous [27]. But  $f$  continuous and  $X$  connected implies  $f(X) = Y$  is connected [27], a contradiction.  $\square$

**Lemma A.2** *Suppose  $\dot{x}(t) = f(x(t))$  with  $f$  globally Lipschitz continuous in  $x$  with constant  $L_f \geq 0$ . Then, for any  $L$  such that  $L \geq L_f$  and  $L > 0$ , and any  $t_2 \geq t_1$ ,*

$$\|x_{t_2} - x_{t_1}\| \leq \|f(x_{t_1})\| L^{-1} (e^{L(t_2-t_1)} - 1).$$

**Proof** Note that for  $t \geq t_1$ ,

$$x_t - x_{t_1} = \int_{t_1}^t f(x_{t_1}) ds + \int_{t_1}^t [f(x_s) - f(x_{t_1})] ds.$$

So that

$$\begin{aligned} \|x_t - x_{t_1}\| &\leq \int_{t_1}^t \|f(x_{t_1})\| ds + \int_{t_1}^t \|f(x_s) - f(x_{t_1})\| ds \\ &\leq (t - t_1) \|f(x_{t_1})\| + \int_{t_1}^t L \|x_s - x_{t_1}\| ds. \end{aligned}$$

Now, substituting  $\tau = t - t_1$  and  $\sigma = s - t_1$ , this becomes

$$\|x_{\tau+t_1} - x_{t_1}\| \leq \tau \|f(x_{t_1})\| + \int_0^\tau L \|x_{\sigma+t_1} - x_{t_1}\| d\sigma$$

Finally, defining  $u(\tau) = \|x_{\tau+t_1} - x_{t_1}\|$ , this becomes

$$u(\tau) \leq \tau \|f(x_{t_1})\| + \int_0^\tau L u(\sigma) d\sigma.$$

The result now follows from the well-known Bellman-Gronwall inequality [14, p. 252].  $\square$

**Corollary A.3** *Under the same assumptions plus the fact that the system was in state  $x_{t_0}$  at time  $t_0 \leq t_1 \leq t_2$ ,*

$$\|x_{t_2} - x_{t_1}\| \leq \|f(x_{t_0})\| L^{-1} (e^{L(t_2-t_0)} - e^{L(t_1-t_0)}).$$

**Proof** Note that Lipschitz continuity gives

$$\|f(x_{t_1})\| \leq L \|x_{t_1} - x_{t_0}\| + \|f(x_{t_0})\|.$$

But, the lemma gives in turn

$$\|x_{t_1} - x_{t_0}\| \leq \|f(x_{t_0})\| L^{-1} (e^{L(t_1-t_0)} - 1).$$

So that the result follows.  $\square$

The following lemma is well-known (see, e.g., [18, p. 169]).

**Lemma A.4** *Let  $y(t), z(t)$  be solutions to  $\dot{x}(t) = f(x(t))$  where  $f$  has global Lipschitz constant  $L \geq 0$ . Then for all  $t \geq t_0$ ,*

$$\|y(t) - z(t)\| \leq \|y(t_0) - z(t_0)\| e^{L(t-t_0)}.$$

$\square$

## References

- [1] P. J. Antsaklis, M. D. Lemmon, and J. A. Stiver, Hybrid system modeling and event identification, Technical Report ISIS-93-002, ISIS Group, University of Notre Dame, 1993.
- [2] E. Asarin and O. Maler, On some relations between dynamical systems and transition systems, preprint, November 1993.
- [3] A. Back, J. Guckenheimer, and M. Myers, A dynamical simulation facility for hybrid systems, in: [16], 255–267.
- [4] C. H. Bennett, Logical reversibility of computation, *IBM Journal of Research and Development*, **6** (1973) 525–532.
- [5] L. S. Bobrow and M. A. Arbib, *Discrete Mathematics*, (W. B. Saunders, Philadelphia, 1974).
- [6] M. S. Branicky, Equivalence of analog and digital computation, Workshop on Continuous Algorithms and Complexity, Centre de Recerca Matematica, Barcelona, Spain (October 1993) abstract.
- [7] M. S. Branicky, Topology of hybrid systems, Proc. 32nd IEEE Conference on Decision and Control, San Antonio, TX (December 1993) 2309–2314.
- [8] M. S. Branicky, Why you can't build an arbiter, Technical Report LIDS-TR-2217, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, December 1993.
- [9] M. S. Branicky, V. S. Borkar, and S. K. Mitter, A unified framework for hybrid control, Proc. 33rd IEEE Conference on Decision and Control, Lake Buena Vista, FL (December 1994) to appear.
- [10] R. W. Brockett, Smooth dynamical systems which realize arithmetical and logical operations, in: H. Nijmeijer and J. M. Schumacher, eds., *Three Decades of Mathematical Systems Theory*, (Springer-Verlag, Berlin, 1989), 19–30.
- [11] R. W. Brockett, Dynamical systems that sort lists, diagonalize matrices, and solve linear programming problems, *Linear Algebra and Its Applications*, **146** (1991) 79–91.
- [12] R. W. Brockett, Hybrid models for motion control systems, in: H. L. Trentelman and J. C. Willems, eds., *Essays in Control: Perspectives in the Theory and its Applications* (Birkhäuser, Boston, 1993) 29–53.
- [13] M. Cosnard, M. Garzon, and P. Koiran, Computability properties of low-dimensional dynamical systems, in: *Proc. 10th Symposium on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science, **665** (Springer-Verlag, New York, 1993).
- [14] C. A. Desoer and M. Vidyasagar, *Feedback Systems: Input-Output Properties*, (Academic Press, New York, 1975).

- [15] R. L. Devaney, *An Introduction to Chaotic Dynamical Systems*, Second Ed., (Addison-Wesley, Redwood City, CA, 1989).
- [16] R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, eds., *Hybrid Systems*, Lecture Notes in Computer Science, **736** (Springer-Verlag, New York, 1993).
- [17] J. Guckenheimer and P. Holmes, *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*, (Springer-Verlag, New York, 1990).
- [18] M. W. Hirsch and S. Smale, *Differential Equations, Dynamical Systems, and Linear Algebra*, (Academic Press, San Diego, 1974).
- [19] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, (Addison-Wesley, Reading, MA, 1979).
- [20] P. Kůrka, Simulation in dynamical systems and Turing machines, Technical report, Department of Mathematical Logic and Philosophy of Mathematics, Charles University, Czech., 1993.
- [21] D. G. Luenberger, *Introduction to Dynamic Systems*, (John Wiley and Sons, New York, 1979).
- [22] G. W. Mackey, *The Mathematical Foundations of Quantum Mechanics*, (Benjamin/Cummings, Reading, MA, 1963).
- [23] L. R. Marino, General theory of metastable operation, *IEEE Transactions on Computers*, **30** (1981) 107–115.
- [24] M. L. Minsky, *Computation: Finite and Infinite Machines*, (Prentice-Hall, Englewood Cliffs, NJ, 1967).
- [25] C. Moore, Unpredictability and undecidability in dynamical systems. *Physical Review Letters*, **64** (1990) 2354–2357.
- [26] C. Moore, Generalized shifts: unpredictability and undecidability in dynamical systems. *Nonlinearity*, **4** (1991) 199–230.
- [27] J. R. Munkres, *Topology*, (Prentice-Hall, Englewood Cliffs, NJ, 1975).
- [28] J. R. Munkres, *Analysis on Manifolds*, (Addison-Wesley, Redwood City, CA, 1990).
- [29] A. Nerode and W. Kohn, Models for hybrid systems: Automata, topologies, controllability, observability, in: [16], 317–356.
- [30] W. Rudin, *Principles of Mathematical Analysis*, Third Ed., (McGraw-Hill, New York, 1976).
- [31] K. S. Sibirsky, *Introduction to Topological Dynamics*, (Noordhoff International Publishing, Leyden, The Netherlands, 1975).
- [32] H. T. Siegelman and E. D. Sontag, Turing computation with neural nets. *Applied Math. Letters*, **4** (1991) 77–80.

- [33] E. D. Sontag, *Mathematical Control Theory*, (Springer-Verlag, New York, 1990).
- [34] L. Tavernini, Differential automata and their discrete simulators, *Nonlinear Analysis, Theory, Methods, and Applications*, **11** (1987) 665–683.
- [35] T. Toffoli, Reversible computing, Technical Report MIT/LCS/TM-151, Laboratory for Computer Science, Massachusetts Institute of Technology, February 1980.
- [36] S. A. Ward and R. H. Halstead, Jr., *Computation Structures*, (MIT Press, Cambridge, MA, 1989).
- [37] N. P. Zhidkov, Nekotore svoistva diskretnikh dinamicheskikh sistem, *Moskovskii Gosudarstvennyi Universitet Lomonosova Uchenye Zapiski*, **163** (1952), *Matematika* **6**, pp. 31–59, (in Russian).