A Survey on Continuous Time Computations

Olivier Bournez[1,2] and Manuel L. Campagnolo[3,4]

[1] INRIA Lorraine
[2] LORIA (UMR 7503 CNRS-INPL-INRIA-Nancy2-UHP), 54506 Vandœuvre-Lès-Nancy, France,
Olivier.Bournez@loria.fr
[3] DM/ISA, Technical University of Lisbon, Tapada da Ajuda, 1349-017 Lisboa, Portugal
[4] SQIG/IT Lisboa
mlc@math.isa.utl.pt

Summary. We provide an overview of theories of continuous time computation. These theories allow us to understand both the hardness of questions related to continuous time dynamical systems and the computational power of continuous time analog models. We survey the existing models, summarizing results, and point to relevant references in the literature.

1 Introduction

Continuous time systems arise as soon as one attempts to model systems that evolve over a continuous space with a continuous time. They can even emerge as natural descriptions of discrete time or space systems. Utilizing continuous time systems is a common approach in fields such as biology, physics or chemistry, when a huge population of agents (molecules, individuals, ...) is abstracted into real quantities such as proportions or thermodynamic data [100], [148].

Several approaches have led to theories on continuous time computations. We will explore in greater depth two primary approaches. One, which we call *inspired by continuous time analog machines*, has its roots in models of natural or artificial analog machinery. The other, which we refer to as *inspired by continuous time system theories*, is broader in scope. It comes from research on continuous time systems theory from a computational perspective. Hybrid systems and automata theory, for example, are two sources.

A wide range of problems related to theories of continuous time computations are encompassed by these two approaches. They originate in fields as diverse as verification (see, e.g., [20]), control theory (see, e.g., [44]), VLSI design (see, e.g., [140],

[141]), neural networks (see, e.g., [160]), and recursion theory on the reals (see, e.g., [145]).

At its beginning, continuous time computation theory was concerned mainly with analog machines. Determining which systems can actually be considered as computational models is a very intriguing question. It relates to the philosophical discussion about what is a programmable machine, which is beyond the scope of this chapter. Nonetheless, some early examples of built analog devices are generally accepted as programmable machines. They include Bush's landmark 1931 Differential Analyzer [50], as well as Bill Phillips's Finance Phalograph, Hermann's 1814 Planimeter, Pascal's 1642 Pascaline, or even the 87 B.C. Antikythera mechanism; see [70]. Continuous time computational models also include neural networks and systems that can be built using electronic analog devices. Since continuous time systems are conducive to modeling huge populations, one might speculate that they will have a prominent role in analyzing massively parallel systems such as the Internet [162].

The first true model of a universal continuous time machine was proposed by Shannon [183], who introduced it as a model of the differential analyzer. During the 1950s and 1960s, an extensive body of literature was published about the programming of such machines.[5] There were also a number of significant publications on how to use analog devices to solve discrete or continuous problems; see, e.g., [200] and the references therein. However, most of this early literature is now only marginally relevant given the ways in which our current understanding of computability and complexity theory have developed.

The research on artificial neural networks, despite the fact that it mainly focused on discrete time analog models, has motivated a change of perspective due to its many shared concepts and goals with today's standard computability and complexity theory [160], [158]. Another line of development of continuous time computation theory has been motivated by hybrid systems, particularly by questions related to the hardness of their verification and control; see, e.g., [44] and [20].

In recent years there has also been a surge of interest in alternatives to classic digital models other than continuous time systems. Those alternatives include discrete-time, analog-space models like artificial neural networks [160], optical models [205], signal machines [76], and the Blum Shub and Smale model [30]. More generally there have also been many recent developments in nonclassical and more-or-less realistic or futuristic models such as exotic cellular automata models [93], molecular or natural computations [96], [3], [122], [163], black hole computations [104], or quantum computations [75], [94], [184], [109]. Some of these contributions are detailed in this volume.

[5] See for example the very instructive Doug Coward's web *Analog Computer Museum* [70] and its bibliography. This literature reveals the quite forgotten art of programming continuous time and hybrid (digital–analog) machines, with a level of sophistication that is close to today's engineering programming.

The computational power of discrete time models are fairly well known and understood thanks in large part to the Church–Turing thesis. The Church–Turing thesis states that all reasonable and sufficiently powerful models are equivalent. For continuous time computation, the situation is far from being so clear, and there has not been a significant effort toward unifying concepts. Nonetheless, some recent results establish the equivalence between apparently distinct models [89], [88], [90], and [35], which give us hope that a unified theory of continuous time computation may not be too far in the future.

This text can be considered an up-to-date version of Orponen's 1997 survey [160]. Orponen states at the end of his introduction that the effects of imprecision and noise in analog computations are still far from being understood and that a robust complexity theory of continuous time models has yet to be developed. Although this evaluation remains largely accurate with regard to imprecision and noise, we will see in the current survey that in the intervening decade much progress in understanding the computability and even the complexity of continuous time computations has been made.

This chapter is organized as follows. In Section 2, we review the most relevant continuous time models. In sections 3 and 4, we discuss, respectively, computability and complexity issues in continuous time computations. In these sections we focus mainly on continuous time dynamical systems. In Section 5, we address the effect of imprecision and noise in analog computations. Finally, in Section 6, we conclude with some general insights and directions for further research in the field of continuous time computation.

2 Continuous Time Models

With a historical perspective in mind, we outline in this section several of the major classes of continuous time models that motivated interest in this field. These models also illustrate concepts like continuous dynamics and input/output.

2.1 Models inspired by analog machines

GPAC and other circuit models

Probably, the best known universal continuous time machine is the *Differential Analyzer*, built at MIT under the supervision of Vannevar Bush [50] for the first time in 1931. The idea of assembling integrator devices to solve differential equations dates back to Lord Kelvin in 1876 [195]. Mechanical,[6] and later on electronic, differential analyzers were used to solve various kinds of differential equations primarily related

[6] And even *MECANO* machines; see [42].

to problems in the field of engineering; see for, e.g., [42], or more generally [204] for historical accounts. By the 1960s, differential analysers were progressively discarded in favor of digital technology.

The first theoretical study of the computational capabilities of continuous time universal machines was published by Shannon. In [183], he proposed what is now referred to as the *General Purpose Analog Computer (GPAC)* as a theoretical model of Vannevar Bush's differential analyzer. The model, later refined in the series of papers [166], [121], [89], [88], consists of families of circuits built with the basic units presented in Figure 1. There are some restrictions to the kinds of interconnectivity that are allowed to avoid undesirable behavior: e.g., nonunique outputs. For further details and discussions, refer to [89] and [87].

Shannon, in his original paper, already mentions that the GPAC generates polynomials, the exponential function, the usual trigonometric functions, and their inverses (see Figure 2). More generally, he claimed in [183] that a function can be generated by a GPAC if and only if it is differentially algebraic; i.e. it satisfies some algebraic differential equation of the form

$$p\left(t, y, y', ..., y^{(n)}\right) = 0,$$

where p is a nonzero polynomial in all its variables. As a corollary, and noting that the Gamma function $\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}dt$ or the Riemann's Zeta function $\zeta(x) = \sum_{k=0}^{\infty} \frac{1}{k^x}$ are not d.a. [175], it follows that the Gamma and the Zeta functions are examples of functions that cannot be generated by a GPAC.

However, Shannon's proof relating functions generated by GPACs with differentially algebraic functions was incomplete (as pointed out and partially corrected by [166], [121]). However, for the more robust class of GPACs defined in [89], the following stronger property holds: a scalar function $f : \mathbb{R} \to \mathbb{R}$ is generated by a GPAC if and only if it is a component of the solution of a system $y' = p(t, y)$, where p is a vector of polynomials. A function $f : \mathbb{R} \to \mathbb{R}^k$ is generated by a GPAC if and only if all of its components are also.

The Γ function is indeed GPAC computable, if a notion of computation inspired from recursive analysis is considered [88]. GPAC computable functions in this sense correspond precisely to computable functions over the reals [35].

Rubel proposed [176] an extension of Shannon's original GPAC. In Rubel's model, the Extended Analog Computer (EAC), operations to solve boundary value problems, or to take certain infinite limits were added. We refer to [140] and [141] for descriptions of actual working implementations of Rubel's EAC.

More broadly, a discussion of circuits made of general basic units has been presented recently in [198]. Equational specifications of such circuits, as well as their semantics, are given by fixed points of operators over the space of continuous streams. Under suitable hypotheses, this operator is contracting and an extension

of Banach fixed point theorem for metric spaces guarantees existence and unicity of the fixed point. Moreover, that fixed point can also be proved to be continuous and *concretely* computable whenever the basic modules also have those properties.

Hopfield network models

Another well-known continuous time model is the "neural network" model proposed by John Hopfield in 1984 in [105]. These networks can be implemented in electrical [105] or optical hardware [193].

A symmetric Hopfield network is made of a finite number, say n, of simple computational units, or *neurons*. The architecture of the network is given by some (nonoriented) graph whose nodes are the neurons and whose edges are labeled by some weights, the *synaptic weights*. The graph can be assumed to be complete by replacing the absence of a connection between two nodes by an edge whose weight is null.

The state of each neuron i at time t is given by some real value $u_i(t)$. Starting from some given initial state $\vec{u}_0 \in \mathbb{R}^n$, the global dynamic of the network is defined by a system of differential equations

$$C_i u_i'(t) = \sum_j W_{i,j} V_j - u_i/R_i + I_i,$$

where $V_i = \sigma(u_i)$, σ is some saturating function such as $\sigma(u) = \alpha \tan u + \beta$, $W_{i,j} = W_{j,i}$ is the weight of the edge between i, and j, C_i, I_i, R_i are some constants [105].

Hopfield proved in [105], by a Lyapunov-function argument, that such systems are *globally asymptotically stable*; i.e., from any initial state, the system relaxes toward some stable equilibrium state. Indeed, consider for example the energy function [105]

$$E = -\frac{1}{2} \sum_i \sum_j W_{i,j} V_i V_j + \sum_i \frac{1}{R_i} \int_0^{V_i} \sigma^{-1}(V) dV + \sum_i I_i V_i.$$

The function E is bounded, and its derivative is negative. Hence the time evolution of the whole system is a motion in a state space that seeks out (possibly local) minima of E.

This convergence behavior has been used by Hopfield to explore various applications such as associative memory or to solve combinatorial optimization problems [105], [106].

An exponential lower bound on the convergence time of continuous time Hopfield networks has been related to their dimension in [188]. Such continuous time symmetric networks can be proved to simulate any finite, binary-state, discrete-time, recurrent neural network [161], [189].

Networks of spiking neurons

If one classifies, following [129], neural network models according to their activation functions and dynamics, three different generations can be distinguished. The first generation, with discontinuous activation functions, includes multilayer perceptrons, Hopfield networks, and Boltzmann machines (see, for example, [2] for an introduction to all mentioned neural network models). The output of this generation of networks is digital. The second generation of networks uses continuous activation functions instead of step or threshold functions to compute the output signals. These functions include feedforward and recurrent sigmoidal neural network, radial basis functions networks, and continuous time Hopfield networks. Their input and output is analog. The third generation of networks is based on spiking neurons and encodes variables in time differences between pulses. This generation exhibits continuous time dynamics and is the most biologically realistic [133].

There are several mathematical models of spiking neurons of which we will focus on one, whose computational properties have been investigated in depth. The Spiking Neural Network model is represented by a finite directed graph. To each node v (neuron) of the graph is associated a *threshold function* $\theta_v : \mathbb{R}^+ \to \mathbb{R} \cup \{\infty\}$, and to each edge (u, v) (synapse) is associated a *response-function* $\epsilon_{u,v} : \mathbb{R}^+ \to \mathbb{R}$ and a *weight-function* $w_{u,v}$.

For a noninput neuron v, one defines its set F_v of *firing times* recursively. The first element of F_v is $\inf\{t|P_v(t) \geq \theta_v(0)\}$, and for any $s \in F_v$, the next larger element of F_v is $\inf\{t|t > s \text{ and } P_v(t) \geq \theta_v(t-s)\}$, where

$$P_v(t) = 0 + \sum_u \sum_{s \in F_u, s<t} w_{u,v}(s)\epsilon_{u,v}(t-s).$$

The 0 above can be replaced by some bias function. We use it here to guarantee that P_v is well defined even if $F_u = \emptyset$ for all u with $w_{u,v} \neq 0$. To approximate biological realism, restrictions are placed on the allowed response-functions and bias-functions of these models; see [129], [130], [153], or [131], [132], for discussions on the model. In particular, rapidly fading memory is a biological constraint that prevents chaotic behavior in networks with a continuous time dynamic. Recently, the use of feedback to overcome the limitations of such a constraint was analyzed in [134].

The study of the computational power of several variants of spiking neural networks was initiated in [126]. Noisy extensions of the model have been considered [127], [128], [135]. A survey of complexity results can be found in [190]. Restrictions that are easier to implement in hardware versions have also been investigated in [137].

$\mathbb{R}$-recursive functions

Moore proposed a theory of recursive functions on the reals in [145], which is defined in analogy with classical recursion theory and corresponds to a conceptual analog

computer operating in continuous time. As we will see, this continuous time model has in particular the capability of solving differential equations, which similar to an idealized analog integrator of the GPAC. In fact, the theory of $\mathbb{R}$-recursive functions can be seen as an extension of Shannon's theory for the GPAC. A general discussion of the motivations behind $\mathbb{R}$-recursion theory can be found in [150].

A function algebra $[B_1, B_2, ...; O_1, O_2, ...]$ is the smallest set containing basic functions $\{B_1, B_2, ...\}$ and is closed under certain operations $\{O_1, O_2, ...\}$, which take one or more functions in the class and create new ones. Although function algebras have been defined in the context of recursion theory on the integers, and has been widely used to characterize computability and complexity classes [62], they are equally suitable to define classes of real-valued recursive functions.

The $\mathbb{R}$-recursive functions were first defined in [145]. These functions are given by the function algebra $\mathcal{M} = [0, 1, U; \mathsf{comp}, \mathsf{int}, \mathsf{minim}]$,[7] where U is the set of projection functions $U_i(\vec{x}) = x_i$, comp is composition, int is an operation that given f and g returns the solution of the initial value problem $h(\vec{x}, 0) = f(\vec{x})$ and $\partial_y h(\vec{x}, y) = g(\vec{x}, y, h)$, and minim returns the smallest zero $\mu_y f(\vec{x}, y)$ of a given f. Moore also studied the weaker algebra $\mathcal{I} = [0, 1, -1, U; \mathsf{comp}, \mathsf{int}]$ and claimed its equivalence with the class of unary functions generated by the GPAC [145].

Many nonrecursively enumerable sets are $\mathbb{R}$-recursive. Since minim is the operation in $\mathcal{M}$ that gives rise to uncomputable functions, a natural question is to ask whether minim can be replaced by some other operation of mathematical analysis. This was done in [149], where minim is replaced by the operation lim, which returns the infinite limits of the functions in the algebra. These authors stratify $[0, 1, -1, U; \mathsf{comp}, \mathsf{int}, \mathsf{lim}]$ according to the allowed number (η) of nested limits and relate the resulting η-hierarchy with the arithmetical and analytical hierarchies. In [124] it is shown that the η-hierarchy does not collapse (see also [123]), which implies that infinite limits and first-order integration are not interchangeable operations [125].

The algebra $\mathcal{I}$ only contains analytic functions and is not closed under iteration [52]. However, if an arbitrarily smooth extension to the reals θ of the Heaviside function is included in the set of basic functions of $\mathcal{I}$, then $\mathcal{I} + \theta$ contains extensions to the reals of all primitive recursive functions.

The closure of fragments of $\mathcal{I} + \theta = [0, 1, -1, \theta, U; \mathsf{comp}, \mathsf{int}]$ under discrete operations like bounded products, bounded sums, and bounded recursion, has been investigated in the thesis [54] and also in the papers [53], [55], [56].

In particular, several authors studied the function algebra

$$\mathcal{L} = [0, 1, -1, \pi, \theta, U; \mathsf{comp}, \mathsf{LI}],$$

[7] We consider that the operator int preserves analyticity (see [52], [55]).

where the LI can only solve *linear* differential equations (i.e., it restricts int to the case $\partial_y h(\vec{x}, y) = g(\vec{x}, y)\, h(\vec{x}, y)$). The class $\mathcal{L}$ contains extensions to the reals of all the elementary functions [53].

Instead of asking which computable functions over $\mathbb{N}$ have extensions to $\mathbb{R}$ in a given function algebra, Bournez and Hainry consider classes of functions over $\mathbb{R}$ computable according to recursive analysis, and they characterize them precisely with function algebras. This was done for the elementarily computable functions [36], characterized as $\mathcal{L}$ closed under a restricted limit schema. This was extended to yield a characterization of the whole class of computable functions over the reals [37], adding a restricted minimisation schema. Those results provide syntactical characterizations of real computable functions in a continuous setting, which is arguably more natural than the higher order Turing machines of recursive analysis.

A more general approach to the structural complexity of real recursive classes, developed in [57], is based on the notion of approximation. This notion was used to lift complexity results from $\mathbb{N}$ to $\mathbb{R}$, and it was applied in particular to characterize $\mathcal{L}$.

Somewhat surprisingly, the results above indicate that two distinct models of computation over the reals (computable analysis and real recursive functions) can be linked in an elegant way.

2.2 Models inspired by continuous time system theories

Hybrid Systems

An increasing number of systems exhibit some interplay between discrete and analog behaviors. The investigation of these systems has led to relevant new results about continuous time computation.

A variety of models have been considered; see, for example, the conference series *Hybrid Systems Computation and Control* or [43]. However, hybrid systems[8] are essentially modeled either as differential equations with discontinuous right-hand sides, as differential equations with continuous and discrete variables, or as hybrid automata. A hybrid automaton is a finite state automaton extended with variables. Its associated dynamics consists of guarded discrete transitions between states of the automaton that can reset some variables. Typical properties of hybrid systems that have been considered are reachability, stability, and controllability.

With respect to the differential equation modeling approach, Branicky proved in [44] that any hybrid system model that can implement a clock and implement general continuous ordinary differential equations can simulate Turing machines. Asarin, Maler,

[8] "Hybrid" refers here to the fact that the systems have intermixed discrete and continuous evolutions. This differs from historical literature about analog computations, where "hybrid" often refers to machines with a mixture of analog and digital components.

and Pnueli proved in [20] that piecewise constant differential equations can simulate Turing machines in $\mathbb{R}^3$, whereas the reachability problem for these systems in dimension $d \leq 2$ is decidable [20]. Piecewise constant differential equations, as well as many hybrid systems models, exhibit the so-called *Zeno's phenomenon*: an infinite number of discrete transitions may happen in a finite time. This has been used in [19] to prove that arithmetical sets can be recognized in finite time by these systems. Their exact computational power has been characterized in terms of their dimension in [32] and [33]. The Jordan's theorem-based argument of [20] to get decidability for planar piecewise constant differential equations has been generalized for planar polynomial systems [60] and for planar differential inclusion systems [22].

There is extensive literature on the hybrid automata modeling approach about determining the exact frontier between decidability and nondecidability for reachability properties, according to the type of allowed dynamics, guards, and resets. The reachability property has been proved decidable for timed automata [5]. By reduction to this result, or by a finite bisimulation argument in the same spirit, this has also been generalized to multirate automata [4], to specific classes of updatable timed automata in [38], [39], and to initialized rectangular automata in [98], [171]. There is a multitude of undecidability results, most of which rely on simulations of Minsky two-counter machines. For example, the reachability problem is semi-decidable but nondecidable for linear hybrid automata [4], [156]. The same problem is known to be undecidable for rectangular automata with at least five clocks and one two-slope variable [98], or for timed automata with two skewed clocks [4]. For discussion of these results, see also [21]. Refer to [28] and [66] or to the survey [29] for properties other than reachability (for example, stability and observability).

O-minimal hybrid systems are initialized hybrid systems whose relevant sets and flows are definable in an o-minimal theory. These systems always admit a finite bisimulation [119]. However, their definition can be extended to a more general class of "nondeterministic" o-minimal systems [46], for which the reachability problem is undecidable in the Turing model, as well as in the Blum Shub Smale model of computation [45]. Upper bounds have been obtained on the size of the finite bisimulation for Pfaffian hybrid systems [116] [117] using the word encoding technique introduced in [46].

Automata theory

There have been several attempts to adapt classical discrete automata theory to continuous time; this is sometimes referred to as the general program of Trakhtenbrot [196].

One attempt is related to timed automata, which can be seen as languages recognizers [6]. Many specific decision problems have been considered for timed automata; see survey [7]. Timed regular languages are known to be closed under intersection, union, and renaming, but not under complementation. The membership and empty

language problems are decidable, whereas inclusion and universal language problems are undecidable. The closure of timed regular languages under shuffling is investigated in [82]. Several variants of Kleene's theorem are established [15], [12], [16], [40], [41], [18]. There have been some attempts to establish pumping lemmas [23]. A review, with discussions and open problems related to this approach, can be found in [10].

An alternative and independent automata theory over continuous time has been developed in [174], [197], and [173]. Here automata are not considered as language recognizers but as computing operators on signals. A signal is a function from the non-negative real numbers to a finite alphabet (the set of the channel's states). Automata theory is extended to continuous time, and it is argued that the behavior of finite state devices is ruled by so-called finite memory retrospective functions. These are proved to be speed-independent, i.e. independent under "stretchings" of the time axis. Closure properties of operators on signals are established, and the representation of finite memory retrospective functions by finite transition diagrams (transducers) is discussed. See also [84] for a detailed presentation of Trakhtenbrot and Rabinovich's theory and for discussions about the representation of finite memory retrospective operators by circuits.

Finally, another independent approach is considered in [182], where Chomsky-like hierarchies are established for families of sets of piecewise continuous functions. Differential equations, associated with specific memory structures, are used to recognize sets of functions. Ruohonen shows that the resulting hierarchies are not trivial and establishes closure properties and inclusions between classes.

2.3 Other computational models

In addition to the two previously described approaches, several other computational models have led to interesting developments in continuous time computation theory.

The question of whether Einstein's general relativity equations admit space-time solutions that allow an observer to view an eternity in a finite time was investigated and proved possible in [104]. The question of whether this implies that super-tasks can in principle be solved has been investigated in [77], [102], [101], [103], [78], [154], [155], and [203].

Some machine-inspired models are neither clearly digital nor analog. For example, the power of planar mechanisms attracted great interest in England and France in the late 1800s and in the 1940s in Russia. Specifically, these consisted of rigid bars constrained to a plane and joined at either end by rotable rivets. A theorem attributed[9] to Kempe [108] states that they are able to compute all algebraic functions; see for, e.g., [9] or [194].

[9] The theorem is very often attributed to Kempe [9], [194], even if he apparently never proved exactly that.

3 ODEs and properties

Most of the continuous time models described above have a continuous dynamics described by differential equations. In Shannon's GPAC and Hopfield networks, the input corresponds to the initial condition, whereas the output is, respectively, the time evolution or the equilibrium state of the system. Other models are language recognizers. The input again corresponds to the initial condition, or some initial control, and the output is determined by some accepting region in the state space of the system. All these systems therefore fall into the framework of dynamical systems.

In this section we will recall some fundamental results about dynamical systems and differential equations and discuss how different models can be compared in this general framework.

3.1 ODEs and dynamical systems

Let us consider that we are working in $\mathbb{R}^n$ (in general, we could consider any vector space with a norm). Let us consider $f : E \to \mathbb{R}^n$, where $E \subset \mathbb{R}^n$ is open. An ODE is given by $y' = f(y)$, and its solution is a differentiable function $y : I \subset \mathbb{R} \to E$ that satisfies the equation.

For any $x \in E$, the fundamental existence-uniqueness theorem (see, e.g., [100]) for differential equations states that if f is Lipschitz on E, i.e., if there exists K such that $||f(y_1) - f(y_2)|| < k||y_1 - y_2||$ for all $y_1, y_2 \in E$, then the solution of

$$y' = f(y), \qquad y(t_0) = x \tag{1}$$

exists and is unique on a certain maximal interval of existence $I \subset \mathbb{R}$. In the terminology of dynamical systems, $y(t)$ is referred to as the *trajectory*, $\mathbb{R}^n$ as the *phase space*, and the function $\phi(t, x)$, which gives the position $y(t)$ of the solution at time t with initial condition x, as the *flow*. The graph of y in $\mathbb{R}^n$ is called the *orbit*.

In particular, if f is continuously differentiable on E, then the existence-uniqueness condition is fulfilled [100]. Most of the mathematical theory has been developed in this case, but it can be extended to weaker conditions. In particular, if f is assumed to be only continuous, then uniqueness is lost, but existence is guaranteed; see, for example, [63]. If f is allowed to be discontinuous, then the definition of the solution needs to be refined. This is explored by Filippov in [81]. Some hybrid system models use distinct and ad hoc notions of solutions. For example, a solution of a piecewise constant differential equation in [20] is a continuous function whose right derivative satisfies the equation.

In general, a dynamical system can be defined as the action of a subgroup $\mathcal{T}$ of $\mathbb{R}$ on a space X, i.e., by a function (a flow) $\phi : \mathcal{T} \times X \to X$ satisfying the following two equations:

$$\phi(0, x) = x, \tag{2}$$

$$\phi(t, \phi(s, x)) = \phi(t + s, x). \tag{3}$$

It is well known that subgroups $\mathcal{T}$ of $\mathbb{R}$ are either dense in $\mathbb{R}$ or isomorphic to the integers. In the first case, the time is called continuous, and in the latter case, discrete.

Since flows obtained by initial value problems (IVP) of the form (1) satisfy equations (2) and (3), they correspond to specific continuous time and space dynamical systems. Although not all continuous time and space dynamical systems can be put in a form of a differential equation, IVPs of the form (1) are sufficiently general to cover a very wide class of such systems. In particular, if ϕ is continuously differentiable, then $y' = f(y)$, with $f(y) = \frac{d}{dt}\phi(t, y)\big|_{t=0}$, describes the dynamical system.

For discrete time systems, we can assume without loss of generality that $\mathcal{T}$ is the integers. The analog of IVP (1) for discrete time systems is a recurrence equation of type

$$y_{t+1} = f(y_t), \qquad y_0 = x. \tag{4}$$

A dynamical system whose space is discrete and that evolves discretely is termed digital; otherwise it is analog. A classification of some computational models according to the nature of their space and time can be found in Figure 3.

3.2 Dissipative and non-dissipative systems

A point x^* of the state space is called an *equilibrium point* if $f(x^*) = 0$. If the system is at x^*, it will remain there. It is said to be *stable* if for every neighborhood U of x^*, there is a neighborhood W of x^* in U such that every solution starting from a point x of W is defined and is in U for all time $t > 0$. The point is *asymptotically stable* if, in addition to the properties above, we have $\lim y(t) = x^*$ [100].

Some local conditions on the differential $Df(x^*)$ of f in x^* have been clearly established. If at an equilibrium point x^* all eigenvalues of $Df(x^*)$ have negative real parts, then x^* is asymptotically stable, and furthermore, nearby solutions approach x^* exponentially. In that case, x^* is called a *sink*. At a stable equilibrium point x^*, no eigenvalue of $Df(x^*)$ can have a positive real part [100].

In practice, Lyapunov's stability theorem applies more broadly (i.e., even if x^* is not a sink). It states that if there exists a continuous function V defined on a neighborhood of x^*, differentiable (except perhaps on x^*) with $V(x^*) = 0$, $V(x) > 0$ for $x \neq x^*$, and $dV(x)/dt \leq 0$ for $x \neq x^*$, then x^* is stable. If, in addition, $dV(x)/dt < 0$ for $x \neq x^*$, then x^* is asymptotically stable; see [100].

If the function V satisfies the previous conditions everywhere, then the system is *globally asymptotically stable*. Whatever the initial point x is, the trajectories will eventually converge to local minima of V. In this context, the Lyapunov function V can be interpreted as an energy, and its minima correspond to attractors of the

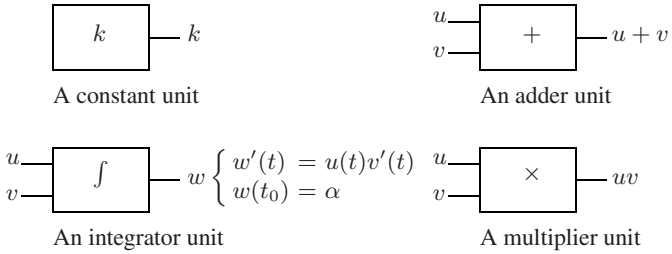

Fig. 1. Different types of units used in a GPAC.

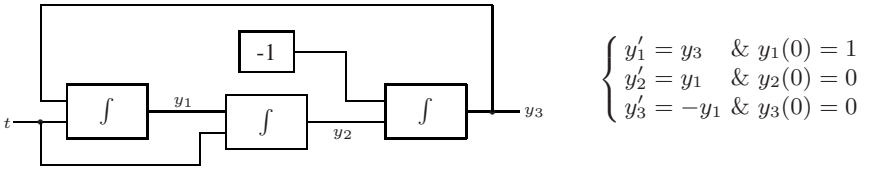

Fig. 2. Generating cos and sin via a GPAC: circuit version on the left and ODE version on the right. One has $y_1 = \cos$, $y_2 = \sin$, and $y_3 = -\sin$.

Space Time	Discrete	Continuous
Discrete	[199] machines [61] lambda calculus [110] recursive functions [164] systems Cellular automata Stack automata Finite state automata ⋮	Discrete time [105] neural networks [186] neural networks [20] PCD systems [31] machines [205] optical machines [76] signal machines [146] dynamical recognizers ⋮
Continuous	[72] BDE models	[183] GPACs Continuous time [105] neural networks [44] hybrid systems [20] PCD systems [5] timed automata [145] $\mathbb{R}$-recursive functions ⋮

Fig. 3. A classification of some computational models, according to their space and time.

dynamical system. These are bounded subsets of the phase space to which regions of initial conditions of nonzero volume converge as time increases.

A dynamical system is called *dissipative* if the volume of a set decreases under the flow for some region of the phase space. Dissipative systems are characterized by the presence of attractors. By opposition, a dynamical system is said to be *volume-preserving* if the volume is conserved. For instance, all Hamiltonian systems are volume-preserving because of Liouville's theorem [8]. Volume-preserving dynamical system cannot be globally asymptotically stable [8].

3.3 Computability of solutions of ODEs

Here we review some results on the computability of solutions of IVPs in the framework of recursive analysis (see, e.g., [201] and the corresponding chapter in this volume).

In general, given a computable function f, one can investigate if the solution of the IVP (1) is also computable in the sense of recursive analysis. If we require that the IVP has a unique solution, then that solution is computable. Formally, if f is computable on $[0,1] \times [-1,1]$ and the IVP $y' = f(t,y)$, $y(0) = 0$ has a unique solution on $[0,b]$, $0 < b \leq 1$, then the solution y is computable on $[0,b]$.

This result also holds for a general n-dimensional IVP if its solution is unique [179]. However, computability of solutions is lost as soon as uniqueness of solutions is relaxed, even in dimension 1. Indeed, the famous result of [167] shows that there exists a polynomial-time computable function $f : [0,1] \times [-1,1] \to \mathbb{R}$, such that the equation $y' = f(t,y)$, with $y(0) = 0$, has nonunique solutions, but none of them is computable on any closed finite time interval.

Similar phenomena hold for other natural equations: the three-dimensional wave equation (which is a partial equation), with computable initial data, can have a unique solution that is nowhere computable[10] [168], [165]. Notice that, even if f is assumed computable and analytic, and the solution unique, it may happen that the maximal interval (α, β) of existence of the solution is noncomputable [92]. This same question is open if f is polynomial. Those authors show, however, that if f and f' are continuous and computable, then the solution of $y' = f(y,t), y(0) = x$, for computable x, is also computable on its maximal interval of existence. Refer also to [169] and [111] for more uncomputability results, and also to [111] and [112] for related complexity issues.

3.4 Static undecidability

As observed in [11] and [181], it is relatively simple but not very informative to get undecidability results with continuous time dynamical systems, if f encodes a

[10] However, in all these cases, the problems under study are ill-posed: either the solution is not unique or it is unstable and the addition of some natural regularity conditions to prevent ill-posedness do yield computability [202].

undecidable problem. To illustrate this, we recall the following example in [181]. Ruohonen discusses the event detection problem: given a differential equation $y' = f(t, y)$, with initial value $y(0)$, decide whether a given condition $g_j(t, y(t), y'(t)) = 0, j = 1, \cdots, k$ happens at some time t in a given interval I. Given the Turing machine $\mathcal{M}$, the sequence $f_0, f_1, \cdots$ of rationals defined by

$$f_n = \begin{cases} 2^{-m} & \text{if } \mathcal{M} \text{ stops in } m \text{ steps on input } n \\ 0 & \text{if } \mathcal{M} \text{ does not stop on input } n \end{cases}$$

is not a computable sequence of rationals, but it is a computable sequence of reals, following the nomenclature of [169]. Now, the detection of the event $y(t) = 0$ for the ordinary differential equation $y' = 0$, given n, and the initial value $y(0) = f_n$, is undecidable over any interval containing 0, because $f_n = 0$ is undecidable.

Another modification can be obtained as follows in [181]. He defines the smooth function

$$g(x) = f_{\lfloor x+1/2 \rfloor} e^{-\tan^2 \pi x},$$

which is computable on $[0, \infty)$. The detection of the event $y_1(t) = 0$ for the ODE

$$\begin{cases} y_1' = g(y_2) - 1 \\ y_2' = 0 \end{cases}$$

given an initial value $y_1(0) = 1$, $y_2(0) = n$, where n is a nonnegative integer is then undecidable on $[0, 1]$.

As put forth in [11], undecidability results given by recursive analysis are somehow built similarly.

3.5 Dynamic undecidability

To be able to discuss in more detail computability of differential equations, we will focus on ODEs that encode the transitions of a Turing machine instead of the result of the whole computation simulation.[11] Typically, we start with some (simple) computable injective function that encodes any configuration of a Turing machine M as a point in $\mathbb{R}^n$. Let x be the encoding of the initial configuration of $\mathcal{M}$. Then, we look for a function $f : E \subset \mathbb{R}^{n+1} \to \mathbb{R}^n$ such that the solution of $y'(t) = f(y, t)$, with $y(0) = x$, at time $T \in N$ is the encoding of the configuration of $\mathcal{M}$ after T steps. We will see, in the remainder of this section, that f can be restricted to have low dimension, to be smooth or even analytic, or to be defined on a compact domain.

Instead of stating that the property above is a Turing machine simulation, we can address it as a reachability result. Given the IVP defined by f and x, and any region $A \subset \mathbb{R}^n$, we are interested in deciding if there is a $t \geq 0$ such $y(t) \in A$, i.e., if the flow starting in x crosses A. It is clear that if f simulates a Turing machine in

[11] This is called dynamic undecidability in [177].

the previous sense, then reachability for that system is undecidable (just consider A as encoding the halting configurations of $\mathcal{M}$). So, reachability is another way to address the computability of ODEs, and a negative result is often a byproduct of the simulation of Turing machines. Similarly, undecidability of event detection follows from Turing simulation results.

Computability of reachable and invariant sets have been investigated in [64] for continuous time systems and in [65] for hybrid systems.

In general, viewing Turing machines as dynamical systems provides them a physical interpretation that is not provided by the von Neumann picture [54]. This also shows that many qualitative features of (analog or nonanalog) dynamical systems, e.g., questions about basins of attraction, chaotic behavior, or even periodicity, are noncomputable [143]. Conversely, this brings into the realm of Turing machines and computability in general questions traditionally related to dynamical systems. These include in particular the relations between universality and chaos [11], necessary conditions for universality [74], the computability of entropy [113], understanding of edge of chaos [120], and relations with the shadowing property [107].

3.6 Embedding Turing machines in continuous time

The embedding of Turing machines in continuous dynamical systems is often realized in two steps. Turing machines are first embedded into analog space, discrete time systems, and then the obtained systems are in turn embedded into analog space and time systems.

The first step can be realized with low-dimensional systems with simple dynamics: [143], [177], [44], [181] consider general dynamical systems, [114] piecewise affine maps, [187] sigmoidal neural nets, [115] closed form analytic maps, which can be extended to be robust [90], and [118] one-dimensional very restricted piecewise-defined maps.

For the second step, the most common technique is to build a continuous time and space system whose discretization corresponds to the embedded analog space discrete time system.

There are several classical ways to discretize a continuous time and space system; see Figure 4. One way is to use a virtual stroboscope: the flow $x_t = \phi(t, x)$, when t is restricted to integers, defines the trajectories of a discrete time dynamical system. Another possibility is through a Poincaré section: the sequence x_t of the intersections of trajectories with, for example, a hypersurface can provide the flow of a discrete time dynamical system. See [100].

The opposite operation, called *suspension*, is usually achieved by extending and smoothing equations, and it usually requires higher dimensional systems. This explains why Turing machines are simulated by three-dimensional smooth continuous time systems in [143], [144] and [44] or by three-dimensional piecewise constant

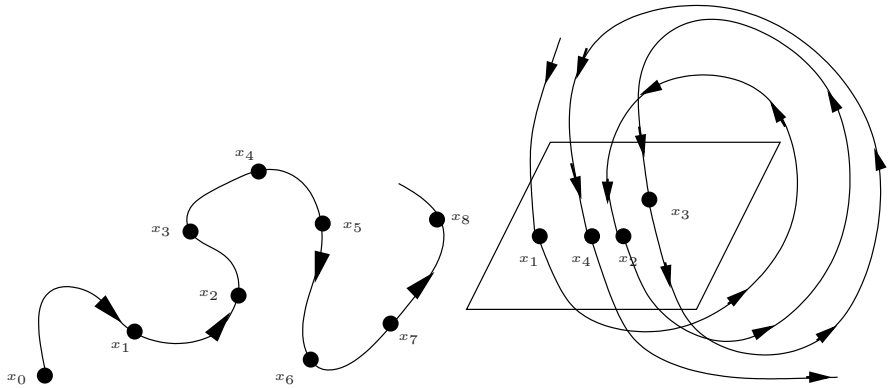

Fig. 4. Stroboscopic map (on left) and Poincaré map (on right) of the dynamic of a continuous time system.

differential equations in [20], while they are known to be simulated in discrete time by only two-dimensional piecewise affine maps in [114]. It is known that two-dimensional piecewise constant differential equations cannot[12] simulate arbitrary Turing machines [20], while the question of whether one-dimensional piecewise affine maps can simulate arbitrary Turing machines is open. Other simulations of Turing machines by continuous time dynamical systems include the robust simulation with polynomial ODEs in [90] and [91]. This result is an improved version of the simulation of Turing machines with real recursive functions in [52], where it is shown that smooth but nonanalytic classes of real recursive functions are closed under iteration. Notice that while the solution of a polynomial ODE is computable on its maximal interval of existence (see Section 3.3), the simulation result shows that the reachability problem is undecidable for polynomial ODEs.

In addition to Turing machines, other discrete models can be simulated by differential equations. Simulating two counter machines can be achieved in two dimensions, or even one dimension, at the cost of a discontinuous ODE [181]. Simulating cellular automata can be done with partial differential equations defined with C^∞ functions [157].

Notice that reversible computations of Turing machines (or counter machines, or register machines) can be simulated by ODEs with backward-unique solutions [177].

Continuous time dynamical systems can in turn be embedded into other continuous time systems. For example, [134] proves that a large class S_n of systems of differential equations are universal for analog computing on time-varying inputs in the following sense: a system of this class can reply to some external input $u(t)$ with the dynamics of any nth order differential equation of the form

[12] See also already mentioned generalizations of this result in [60] and [22].

$z^{(n)}(t) = G(z(t), z'(t), \cdots, z^{(n-1)}(t)) + u(t)$, if a suitable memoryless feedback and readout functions are added. As the nth order differential equation above can simulate Turing machines, systems from S_n have the power of a universal Turing machine. But since G is arbitrary, systems from S_n can actually simulate any conceivable continuous dynamic response to an input stream. Moreover, this results holds for the case where inputs and outputs are required to be bounded.

3.7 Discussion issues

The key technique in embedding the time evolution of a Turing machine in a flow is to use "continuous clocks" as in [44].[13]

The idea is to start from the function $f : \mathbb{R} \to \mathbb{R}$, preserving the integers, and build the ordinary differential equation over $\mathbb{R}^3$

$$\begin{aligned} y_1' &= c(f(r(y_2)) - y_1)^3\theta(\sin(2\pi y_3)),\\ y_2' &= c(r(y_1) - y_2)^3\theta(-\sin(2\pi y_3)),\\ y_3' &= 1. \end{aligned}$$

Here $r(x)$ is a rounding-like function that has value n whenever $x \in [n - 1/4, n + 1/4]$ for some integer n, and $\theta(x)$ is 0 for $x \leq 0$, $\exp(-1/x)$ for $x > 0$, and c is some suitable constant.

The variable $y_3 = t$ is the time variable. Suppose $y_1(0) = y_2(0) = x \in \mathbb{N}$. For $t \in [0, 1/2]$, $y_2' = 0$, and hence y_2 is kept fixed to x. Now, if $f(x) = x$, then y_1 will be kept to x. If $f(x) \neq x$, then $y_1(t)$ will approach $f(x)$ on this time interval, and from the computations in [54], if a large enough number is chosen for c we can be sure that $|y_1(1/2) - f(x)| \leq 1/4$. Consequently, we will have $r(y_1(1/2)) = f(x)$. Now, for $t \in [1/2, 1]$, roles are inverted: $y_1' = 0$, and hence y_1 is kept fixed to the value $f(x)$. On that interval, y_2 approaches $f(x)$, and $r(y_2(1)) = f(x)$. The equation has a similar behavior for all subsequent intervals of the form $[n, n + 1/2]$ and $[n+1/2, n+1]$. Hence, at all integer time t, $f^{[t]}(x) = r(y_1(t))$.[14] [124] proposes a similar construction that returns $f^{[\lfloor t \rfloor]}(x)$ for all $t \in \mathbb{R}$.

In other words, the construction above transforms a function over $\mathbb{R}$ into a higher dimensional ordinary differential equation that simulates its iterations. To do so, $\theta(\sin(2\pi y_3))$ is used as a kind of clock. Therefore, the construction is essentially "hybrid" since it combines smooth dynamics with nondifferentiable, or at least non-analytic clocks to simulate the discrete dynamics of a Turing machine. Even if the flow is smooth (i.e., in C^∞) with respect to time, the orbit does not admit a tangent at every point since y_1 and y_2 are alternatively constant. Arguably, one can overcome this limitation by restricting Turing machine simulations to analytic flows and maps.

[13] Branicky attributes the idea of a two phase computation to [47] and [48]. A similar trick is actually present in [177]. We will actually not follow [44] but its presentation in [54].

[14] $f^{[t]}(x)$ denotes the tth iteration of f on x.

Although it was shown that analytic maps over unbounded domains are able to simulate the transition function of any Turing machine in [115], only recently it was shown that Turing machines can be simulated with analytic flows over unbounded domains in [90]. It would be desirable to extend the result to compact domains. However, it is conjectured in [147] that this is not possible, i.e., that no analytic map on a compact finite-dimensional space can simulate a Turing machine through a reasonable input and output encoding.

3.8 Time and space contractions

Turing machines can be simulated by ODEs in real time: for example, in the constructions we described above, the state $y(T)$ at time $T \in N$ of the solution of the ordinary differential equation encodes the state after T steps of the Turing machine. However, since continuous time systems might undergo arbitrary space and time contractions, Turing machines, as well as accelerating Turing machines[15] [71], [67], [68] or even oracle Turing machines, can actually be simulated in an arbitrary short time.

In the paragraphs below, we will follow Ruohonen [177] who denotes a continuous time system by the triplet (F, n, A), where F defines the ordinary differential equation $y' = F(y)$ over $\mathbb{R}^n$, with accepting set A: some input x is accepted iff the trajectory starting with initial condition x crosses A.

A machine $\mathcal{M} = (F, n, A)$ can be accelerated: the substitution $t = e^u - 1$ for instance changes $\mathcal{M}$ to $((G, 1), n + 1, A \times \mathbb{R})$, where

$$\frac{dg}{du} = G(g(u), u) = F(g(u))e^u \text{ and } g(u) = y(e^u - 1),$$

yielding an exponential time acceleration. Note that the derivatives of the solution with respect to the new time variable u are exponentially larger. Furthermore, the substitution $t = \tan(\pi u/2)$ gives an infinite time acceleration, i.e., compresses any computation, even an infinite one, into the finite interval $0 \leq u < 1$. Now, the derivatives go to infinity during the course of computation.

Turning to space contraction, replacing the state $y(t)$ of the machine $\mathcal{M} = (F, n, A)$ by $r(t) = y(t)e^{-t}$ gives an exponentially downscaled machine $((H, 1), m + 1, H_1)$ where

$$\frac{dr}{dt} = H(r(t), t) = F(r(t)e^t)e^{-t} - r(t)$$

and

$$H_1 = \{(e^{-t}q, t) | q \in A \text{ and } t \geq 0\}.$$

Obviously, this transformation reduces exponentially the distance between trajectories, which require increased precision to be distinguished.

[15] Similar possibilities of simulating accelerating Turing machines through quantum mechanics are discussed in [51].

Hardness results in the levels of the arithmetical or analytical hierarchy for several decision problems about continuous time systems are derived from similar constructions in [177], [178], [145], and [19]. Completeness results, as well as exact characterizations of the recognition power of piecewise constant derivative systems, according to their dimensions have been obtained in [32] and [33]. Notice that such phenomena are instances of the so-called Zeno's phenomena in hybrid systems literature: [5] and [19].

It can be observed that previous constructions yield undecidability results only for functions over infinite or half-open intervals, since positive reals, corresponding to Turing machines integer time, are mapped to intervals of the form $[0, 1)$. An analytical construction is indeed possible over a finite closed domain of the form $[0, 1]$, with a function G that is continuous and bounded on $[0, 1]$, but nondifferentiable. It follows that the event detection problem, for example, is undecidable even with continuous functions over compact intervals [180].

Undecidability is ruled out, however, if the function G is sufficiently smooth (say, in $\mathcal{C}^1$), if both G and the initial value are computable, and if a sufficiently robust acceptance condition is considered. Indeed, problems such as the event detection problem then become decidable, since the system can be simulated effectively [180].

Instead of embedding Turing machines into continuous dynamical systems, it is natural to ask whether there is a better way to think about computation and complexity for the dynamical systems that are commonly used to model the physical world. We address this issue in the next section.

4 Toward a complexity theory

Here we discuss several different views on the complexity of continuous dynamical systems. We consider general systems and question the difficulty of simulating continuous time systems with a digital model. We then focus on dissipative systems, where trajectories converge to attractors. In particular, we discuss the idea that the computation time should be the natural time variable of the ODE. Finally, we review complexity results for more general continuous time systems that correspond to classes of real recursive functions.

4.1 General continuous dynamical systems

In [200] it was asked whether analog computers can be more efficient than digital ones. Vergis et al. also postulated the "Strong Church's Thesis," which states that the time required by a digital computer to simulate any analog computer is bounded by a polynomial function of the resources used by the analog computer. They claim that

the Strong Church's Thesis is provably true for continuous time dynamical systems described by any Lipschitzian ODE $y' = f(y)$.

The resources used by an analog computer include the time interval of operation, say $[0, T]$, the size of the system, which can measured by $\max_{t\in[0,T]} ||y(t)||$, as well as the bound on the derivatives of y. For instance, mass, time of operation, maximum displacement, velocity, acceleration, and applied force are all resources used by a particle described by Newtonian mechanics [200].

The claim above depends on the definition of "simulation." In the article [200] it is considered that the IVP $y' = f(y)$, $y(0) = x$ is simulated if, given T and some precision ε, one can compute an approximation of $y(T)$ with a margin of error of at most ε. Using Euler's method to solve this problem, and considering that the round-off error is less than σ, the total error bound is given by

$$||y(T) - y_N^*|| \leq \frac{h}{\lambda}\left[\frac{R}{2} + \frac{\sigma}{h2}\right](e^T\lambda - 1), \tag{5}$$

where y_N^* is the approximation after N steps, h is the step size, λ is the Lipschitz constant for f on $[0, T]$, and $R = \max\{||y''(t)||, t \in [0, T]\}$. From the bound in (5), Vergis et al. conclude that the number N of necessary steps in Euler's method is polynomial in R and $\frac{1}{\varepsilon}$. They use this fact to claim that the Strong Church's Thesis is valid for ODEs. However, N is exponential in T, which is the time of operation of this analog computer. This makes the argument in [200] inconclusive, as pointed out in [160].

More recently, Smith discusses in [192] if hypercomputation is possible with respect to the n-body problem in mechanics. In particular, he shows that the exponential dependence in T can be eliminated. As observed in [192], all classical numerical methods of fixed degree for solving differential equations suffer from the same exponential dependence in T. However, by considering a combination of Runge–Kutta methods with degrees varying linearly with T, it is possible to derive a method that only requires N to be polynomial in T, as long as the absolute value of each component of f, y, and the absolute value of each partial derivative of f with respect to any of its arguments, having total differentiation degree k, is in $(kT)^{\mathcal{O}(k)}$ [192]. The implications of these results for Strong Church's Thesis are discussed in [192] and [34].

The same question can be addressed in the framework of recursive analysis. When $f : [0, 1] \times [-1, 1] \to \mathbb{R}$ is polynomial time computable and satisfies a weak form of the Lipschitz condition, the unique solution y on $[0, 1]$ of IVP $y' = f(t, y)$, $y(0) = 0$ is always polynomial space computable [111]. Furthermore, solving in polynomial time a differential equation with this weak Lipschitz condition is essentially as difficult as solving a PSPACE-complete problem, since there exists a polynomial time computable function f as above whose solution y is not polynomial time computable unless $P = PSPACE$ [111], [112].

Ko's results are not directly comparable with the polynomial bound shown in [192]. In recursive analysis, the input's size is the number of bits of precision. If the bound on the error of the approximation of $y(t)$ is measured in bits; i.e., if $\varepsilon = 2^{-d}$, then the required number of steps N in [192] is exponential in d.

If f is analytic, then the solution of $y' = f(y)$ is also analytic. In that case, timestepping methods can be avoided. That is the approach followed in [142], where it is proved using recursive analysis that if f is analytic and polynomial time computable, then the solution is also polynomial time computable.

In short, although Strong Church's Thesis holds for analytic ODEs, it has not yet been fully proved for general systems of Lipschitzian ODEs. Hence, the possibility of super-polynomial computation by differential equations cannot be ruled out, at least in principle. For informal discussions on Strong Church's Thesis, refer to [1] and [139].

Several authors have shown that certain decision or optimization problems (e.g., graph connectivity or linear programming) can be solved by specific continuous dynamical systems. Examples and references can be found in the papers [191], [200], [48], [79], [97] and [27].

4.2 Dissipative systems

We now focus on dissipative systems and review two approaches. The first is about neural network models, such as continuous Hopfield networks, that simulate circuits in a nonuniform manner, and leads to lower bounds on the complexity of such networks. The second deals with convergence to attractors and considers suitable energy functions as ways to measure the complexity of continuous time processes.

When considering dissipative systems, such as Hopfield neural networks, the following approach to a complexity theory is natural. Consider families $(C_n)_{n\in N}$ of continuous time systems, for each input length $n \geq 0$. Given some digital input $w \in \{0,1\}^*$, the system C_n evolves on input w (or some encoding of w), where n is the length of w. It will eventually reach some stable state, which is considered the result of the computation.

This circuit inspired notion of computability is the most common in the literature about the computational complexity of neural networks models; see survey [190]. With respect to this approach, continuous time symmetric Hopfield networks with a saturated linear activation function have been proved to simulate arbitrary discrete-time binary recurrent neural networks, at the cost of only a linear size overhead [161], [189]. This might be thought counterintuitive, since such symmetric networks, which are constrained by some Liapunov energy function, can only exhibit convergence phenomena, and hence cannot even realize a simple alternating bit. However, the convergence of dissipative systems can be exponentially long in the size of the

system [188], and hence the simulation can be accomplished using a subnetwork that provides 2^n clock pulses before converging.

The languages recognized by polynomial size families of discrete-time Hopfield networks have been proved in [159] to correspond to nonuniform complexity class $PSPACE/poly$ for arbitrary interconnection weights, and to $P/poly$ for polynomially bounded weights. Therefore, families of continuous time symmetric Hopfield networks have the same lower bounds. However, these lower bounds may be not tight, since upper bounds for continuous time dynamics are not known [189], [190].

Let us now turn our attention to dissipative systems with a Lyapunov function E.

Gori and Meer [86] consider a computational model that has the capability of finding the minimizers (i.e., the points of local or global minimum) of E. To prevent the complexity of a problem from being hidden in the description of E, this function must be easy to compute. In that setting, a problem Π is considered easy if there exists a unimodal function E (i.e., all local minimizers of E are global minimizers) such that the solution of Π can be obtained from the global minimum of E.

More precisely, Gori and Meer investigate in [86] a model where a problem Π over the reals is considered to be solved if there exists a family $(E_n)_n : \mathbb{R}^n \times \mathbb{R}^{q(n)} \to \mathbb{R}$ of energy functions, given by a uniform family of straight line programs (q is some fixed polynomial), and another family $(N_n)_n$ of straight line programs, such that for all input d, a solution $\Pi(d)$ of the problem can be computed using $N_{q(n)}(w^*)$, from a global minimizer w^* of $w \to E_n(d, w)$.

Gori and Meer define classes U and NU in analogy with P and NP in classical complexity. U corresponds to the above-mentioned case where for all d, $w \to E_n(d, w)$ is unimodal, in opposition to NU where it needs not be unimodal. Notions of reductions are introduced, and it is proved that the natural optimization problem "find the minimum of some linear objective function over a set defined by quadratic multivariate polynomial constraints" is NU-hard. They show that there exist (artificial) NU complete problems. These ideas are generalized to obtain a polynomial hierarchy, with complete problems [86].

Actually, Gori and Meer's proposed framework is rather abstract, avoiding several problems connected to what one might expect of a true complexity theory for continuous time computations. Nonetheless, it has the great advantage of not relying on any particular complexity measure for the computation of trajectories. See the interesting discussion in [86].

However, one would like to understand the complexity of approaching the minima of energy functions, which correspond to the equilibria of dynamical systems. First steps toward this end have been investigated in [27], where dissipative systems with exponential convergence are explored. Recall that if x^* is a sink, then the rate of convergence toward x^* satisfies

$$|x(t) - x^*| \equiv e^{-\lambda t},$$

where $-\lambda$ is the largest real part of an eigenvalue of $Df(x^*)$. This means that $\tau = 1/\lambda$ is a natural characteristic time of the attractor: every $\tau \log 2$ time units, a new bit of the attractor is computed.

For the systems considered in [27], each sink has an attracting region, where the trajectories are trapped. One can define the computation time t_c of a dissipative continuous time dynamical system as $t_c = \max(t_c(\epsilon), t_c(U))$, where $t_c(\epsilon)$ is the time required to reach some ϵ vicinity of some attractor, and $t_c(U)$ is the time required to reach its attracting region. Then, $T = \frac{t_c}{\tau}$ is a dimensionless complexity measure, invariant under any linear time contraction.

Two continuous time algorithms, MAX to compute the maximum of n numbers, and $FLOW$ to compute the maximum flow problem have been studied in this framework in [27]. MAX has been shown to belong to proposed complexity class $CLOG$ (continuous log time) and $FLOW$ to CP (continuous polynomial time). The authors conjecture that CP corresponds to classical polynomial time [27]. Both MAX and $FLOW$ algorithms are special cases of a flow proposed in [80] to solve linear programming problems, which are further investigated in [24] and [25]. Variations on definitions of complexity classes, as well as ways to introduce nondeterministic classes in relation to computations by chaotic attractors, have also been discussed in [185].

4.3 Complexity and real recursive functions

Real recursive functions are a convenient way to analyze the computational power of certain operations over real functions. Additionally, given a continous time model, if its equivalence with a function algebra of real recursive functions can be established, then some properties of the model can be proved inductively over the function algebra.

Since many time and space complexity classes have recursive characterizations over $\mathbb{N}$ [62], structural complexity results about discrete operations may imply lower and upper bounds on the computational complexity of real recursive functions. This approach was followed in [53] to show that $\mathcal{L}$ contains extensions of the elementary functions, and it was further developed in [56] to obtain weaker classes that relate to the exponential space hierarchy. This tells us something about the computational complexity of certain dynamical systems. For instance, $\mathcal{L}$ corresponds to cascades of finite depth, each level of which depends linearly on its own variables and the output of the level before it.

Results about the idea of lifting computability questions over $\mathbb{N}$ to $\mathbb{R}$ have been discussed before. Concerning complexity, the question $P = NP$ in classical complexity has been investigated using real recursive functions by Costa and Mycka. In particular, they propose two classes of real recursive functions such that their inequality would imply $P \neq NP$ in [69] and [151]. More generally a part of Costa and Mycka's program, which is explicitly stated in [150] and [152], uses recursion theory

on the reals to establish a bridge between computability and complexity theory and mathematical analysis.

5 Noise and Robustness

Up to this point we have considered continuous time computations in idealized, noise-free spaces. As was also the case in the survey by Orponen [160], most of the results we discussed disregard the impact of noise and imprecision in continuous time systems. This is a recurrently criticized weakness of the research in the field. Although there have not been major breakthroughs with regard to these problems as they relate specifically to continuous time computations, some interesting developments concerning noise and imprecision have come about in discrete time analog computation studies. In this section we will broaden our scope to discuss a number of discrete time results. We believe that some of these studies and results might be generalized to, or at least provide some insight into, the effects of noise and imprecision on continuous time systems, although this work has yet to be done.

We first focus on systems with a bounded state space about which a folklore conjecture claims that robustness implies decidability. We review some results that support this conjecture as well as others that challenge it. At the end of this section, we discuss continuous time systems with unbounded state spaces.

Common techniques to simulate Turing machines by dynamical systems in bounded state spaces require the encoding of the configuration of the Turing machine into real numbers. Since Turing machines have unbounded tapes (otherwise they would degenerate into finite automata), these simulations are destroyed if the real numbers or the functions involved are not represented with infinite precision. This leads to the folklore conjecture, popular in particular in the verification community, which states that undecidability do not hold for "realistic," "unprecise," "noisy," "fuzzy," or "robust" systems. See, for example, [85] and [83] for various statements of this conjecture and [13] for discussions on other arguments that lead to this conjecture.

There is no consensus on what is a realistic noise model. A discussion of this subject would require to question what are good models of the physical world. In the absence of a generally accepted noise model, one can however consider various models for noise, imprecision, or smoothness conditions, and one can investigate the properties of the resulting systems.

In particular, there have been several attempts to show that noisy analog systems are at best equivalent to finite automata. Brockett proved that continuous time dynamical systems can simulate arbitrary finite automata in [47]. Using topological arguments based on homotopy equivalence relations and associated Deck transformations, he showed in [49] that some automata can be associated with dissipative continuous time systems.

Maass and Orponen proved that the presence of bounded noise reduces the power of a large set of discrete time analog models to that of finite automata in [136]. This extends a previous result established in [58] and [59] for the case where the output is assumed to be perfectly reliable (i.e., $\rho = 1/2$ in what follows).

Maass and Orponen's idea is to replace a perfect discrete time dynamic of type $x_{i+1} = f(x_i, a_i)$, where a_i is the symbol input at time i, over a compact domain, by a probabilistic dynamic

$$\text{Probability } (x_{i+1} \in B) = \int_{q \in B} z(f(x_i, a_i), q) d\mu, \tag{6}$$

where B is any Borel set. Here, z is a density kernel reflecting arbitrary noise, which is assumed to be piecewise equicontinuous. This means that, for all ϵ, there exists δ such that for all r, p, q, $\|p - q\| \leq \delta$ implies $|z(r, p) - z(r, q)| \leq \epsilon$. They denote by $\pi_x(q)$ the distribution of states after string x is processed from some fixed initial state q, and they consider the following robust acceptance condition: a language L is recognized, if there exists $\rho > 0$ such that $x \in L$ iff $\int_F \pi_{xu}(q) d\mu \geq 1/2 + \rho$ for some $u \in \{U\}^*$, and $x \notin L$ iff $\int_F \pi_{xu}(q) d\mu \leq 1/2 - \rho$ for all $u \in \{U\}^*$, where U is the blank symbol, and F is the set of accepting states. Then, they show that the space of functions $\pi_x(.)$ can be partitioned into finitely many classes C such that two functions $\pi_x(.)$ and $\pi_y(.)$ in the same class satisfy $\int_r |\pi_x(r) - \pi_y(r)| d\mu \leq \rho$. Therefore, two words x, y in the same class satisfy $xw \in L$ iff $yw \in L$ for all words w.

In fact, for any common noise, such as Gaussian noise, which is nonzero on a sufficiently large part of the state space, systems described by (6) are unable to recognize even arbitrary regular languages [138]. They recognize precisely the definite languages introduced by [172], as shown in [138] and [26]. If the noise level is bounded, then all regular languages can be recognized [136]. Feedback continuous time circuits in [134] have the same computational power when subject to bounded noise.

As an alternative to the probabilistic approach of Maass and Orponen, noise can be modeled through nondeterminism. One can associate with deterministic noise-free discrete time dynamical system S defined by $x_{i+1} = f(x_i)$, the nondeterministic ϵ-perturbated system S_ϵ whose trajectories are sequences $(x_n)_n$ with $\|x_{i+1} - f(x_i)\| \leq \epsilon$. For a dynamical system S, it is natural to consider the predicate Reach$[S](x, y)$ (respectively, Reach$_n[S](x, y)$), which is true if there exists a trajectory of S from x to y (resp. in $i \leq n$ steps). Then, algorithmic verification of safety of state properties is closely related to the problem of computing reachable states. Given S, and a subset of initial states S_0, let Reach$[S]$ denote the set of y's such that Reach$[S](x, y)$ for some $x \in S_0$. Given a state property p (i.e., a property that is either true or false in a state s), let $[[\neg p]]$ denote the subset of states s where p is false. Then S is safe (p is an invariant) iff Reach$[S] \cap [[\neg p]] = \emptyset$ (see, for example, [4] and [156]).

If the class of systems under consideration is such that relation $\text{Reach}_n[S](x, y)$ is recursive[16] (assuming that S_0 recursively enumerable), then $\text{Reach}[S]$ is recursively enumerable because $\text{Reach}[S] = \bigcup_n \text{Reach}_n[S]$. Several papers have been devoted to prove that $\text{Reach}[S]$ is indeed recursive for classes of dynamical systems under different notions of robustness. We now review several of them.

Fränzle observes in [85] that the computation of $\text{Reach}[S_\epsilon]$ by $\text{Reach}[S_\epsilon] = \bigcup_n \text{Reach}_n[S_\epsilon]$ must always terminate if $\text{Reach}[S_\epsilon]$ has a strongly finite diameter. This means that there exists an infinite number of points in $\text{Reach}[S_\epsilon]$ at a mutual distance of at least ϵ, which is not possible over a bounded domain. It follows that if we call robust a system that is either unsafe or whose ϵ-perturbated system is safe for some ϵ, then safety is decidable for robust systems over compact domains [85].

Consider as in [170] the relation $\text{Reach}_\omega[S] = \bigcap_{\epsilon>0} \text{Reach}[S_\epsilon]$, corresponding to states that stay reachable when noise converges to 0. Asarin and Bouajjani prove in [14] that for large classes of discrete and continuous time dynamical systems (Turing machines, piecewise affine maps, piecewise constant differential equations), $\text{Reach}_\omega[S]$ is co-recursively enumerable. Furthermore, any co-recursively enumerable relation is of form $\text{Reach}_\omega[S]$ for some S for the classes that Asarin and Bouajjani consider. Therefore, if we call robust a system such that $\text{Reach}[S] = \text{Reach}_\omega[S]$, then computing $\text{Reach}[S]$ is decidable for robust systems.

Asarin and Collins considered in [17] a model of Turing machines exposed to a small stochastic noise, whose computational power have been characterized to correspond to Π_2^0. It is interesting to compare this result with previous results where a small nondeterministic noise lead to Π_1^0 (co-recursively enumerable sets) computational power only.

We now turn our attention to results that challenge the conjecture that robustness implies decidability. A first example is that the safety of a system is still undecidable if the transition relation of the system is open, as proved in [99], and [13]. However, the question for the restriction to a uniform nondeterministic noise bounded from below is open [13].

Noise can also be modeled by perturbating trajectories. Gupta, Henzinger, and Jagadeesan consider in [95] a metric over trajectories of timed automata, and assume that if a system accepts a trajectory, then it must accept neighboring trajectories also. They prove that this notion of robustness is not sufficient to avoid undecidability of complementation for Timed automata. Henzinger and Raskin prove in [99] that major undecidability results about verification of hybrid systems are still undecidable for robust systems in that sense.

Finally, we review a recent robustness result for continuous time dynamical systems with unbounded state space. Graça, Campagnolo, and Buescu prove in [91] that polynomial differential equations can simulate robustly Turing machines in real time. More precisely, let us consider that $\theta : \mathbb{N}^3 \to \mathbb{N}^3$ is the transition function

[16] Recursive in x, y, and n.

some Turing machine M whose configuration is encoded on $\mathbb{N}^3$. Then, there is a $\epsilon > 0$, a solution f of a polynomial ODE, and an initial condition $f(0)$ such that the solution of $y' = f(t, y)$ encodes the state of M after t steps with error at most ϵ. Moreover, this holds for a neighborhood of any integer t even if f and the initial condition $f(0)$ are perturbed. Obviously, this kind of simulation requires the system to have an unbounded state space.

6 Conclusion

Having surveyed the field of continuous time computation theory, we see that it provides insights into many diverse areas such as verification, control theory, VLSI design, neural networks, analog machines, recursion theory, theory of differential equations, and computational complexity.

We have attempted to give a systematic overview of the most relevant models and results on continuous time computations. In the last decade many new results have been obtained, indicating that this is an active field of research. We reviewed recent developments of the theory of continuous time computation with respect to computability, complexity, and robustness to noise, and we identified several open problems. To conclude, we will discuss some directions for future research related to these areas.

Computability

It is not clear whether a unifying concept similar to the Church–Turing thesis exists for continuous time computation. Although it has been shown that some continuous time models exhibit super Turing power, these results rely on the use of an infinite amount of resources such as time, space, precision, or energy. In general, it is believed that "reasonable" continuous time models cannot compute beyond Turing machines. This raises the question if physically realistic continuous time computation can be as powerful as digital computation. We saw that if we restrict continuous time systems to evolve in a bounded state space and to be subjected to noise, then they become comparable with finite automata. However, with a bounded state space, Turing machines also degenerate into finite automata. Since analytic and robust continuous time systems can simulate Turing machines in an unbounded state space, we believe that digital computation and analog continuous time computation are equally powerful from the computability point of view. Moreover, as we saw, several recent results establish the equivalence between functions computable by polynomial ODEs, GPAC-computable functions and real computable functions in the framework of recursive analysis. These kind of results reinforce the idea that there could be an unified framework for continuous time computations, analogous to what occurs in classical computation theory.

We feel that a general paradigm of realistic continuous time computations ideally should only involve analytic functions, since these are often considered as the most acceptable from a physical point of view. Continuous dynamical systems are a natural form of representing continuous time processes. Classic systems like the van der Pol equation, the Lotka–Volterra system, or the Lorenz equations are described with differential equations with an analytic, even polynomial, right-hand side. These physics-related arguments combined with the computability properties of systems of polynomial differential equations lead us to suggest that this continuous time model is a possible candidate for a general paradigm of continuous time computation. We believe that this idea deserves further investigation.

Complexity

We saw that a complexity theory for continuous time computation is still under way and that there has not been an agreement between authors on basic definitions such as computation time or input size. The results described in Section 4 are either derived from concepts that are intrinsic to the continuous time systems under study or related to classical complexity theory. As computable analysis is a well-established and understood framework for the study of computational complexity of continuous time systems, we believe that understanding relations between different approaches and computable analysis from a complexity point of view is of first importance. There are still many open questions about upper bounds for continuous time models. For example, upper bounds are not known for Hopfield networks and general systems of Lipschitizian ODEs, which compromises the validity of the Strong Turing thesis. We saw that this thesis might hold for systems of analytic ODEs. This leads us to ask whether a continuous time computation theory based on polynomial ODEs could be naturally extended to a complexity theory.

Computable analysis also permits study of the complexity of real recursive functions. One of the most intriguing areas of research in continuous time computation tries to explore the link between real recursive functions and computational complexity to establish a translation of open problems of classic complexity into analysis.

Robustness

We saw that very little research has been done with respect to the robustness and tolerance to noise of continuous time systems. One might ask how the power of analog computations increases with their precision. This question was raised and formalized for discrete time analog systems, in particular for dynamical recognizers, in [147], but most of the research in that direction has yet to be done. Many interesting open questions arise if one asks whether undecidability results for continuous time systems still hold for robust systems. This is of first importance for example for the verification of hybrid systems, since this question is closely related to the question

of termination of automatic verification procedures. A better understanding of the hypotheses under which noise yield decidability or undecidability is required. For example, nondeterministic noise on open systems does not rule out undecidability, but the question is unanswered for a uniform noise bounded from below [13].

Acknowledgments. We would like to thank all our colleagues in a wide sense, since this survey benefited from recent and old discussions with a long list of people. Many of them have their work cited in the text. We would also like to deeply thank Kathleen Merrill for her careful reading of the text and for her suggestions to improve its clarity and an anonymous referee for his/her helpful advice. This work was partially supported by EGIDE and GRICES under the Program *Pessoa* through the project *Calculabilité et complexité des modèles de calculs à temps continu*, by *Fundação para a Ciência e a Tecnologia* and FEDER via the Center for Logic and Computation - CLC and the project ConTComp POCTI/MAT/45978/2002.

References

1. Aaronson, S. (2005). NP-complete problems and physical reality. *ACM SIGACT News*, 36(1):30–52.
2. Abdi, H. (1994). A neural network primer. *Journal of Biological Systems*, 2:247–281.
3. Adleman, L. M. (1994). Molecular computation of solutions to combinatorial problems. *Science*, 266:1021–1024.
4. Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T. A., Ho, P. H., Nicollin, X., Olivero, A., Sifakis, J., and Yovine, S. (1995). The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34.
5. Alur, R. and Dill, D. L. (1990). Automata for modeling real-time systems. In Paterson, M., editor, *Automata, Languages and Programming, 17th International Colloquium, ICALP90, Warwick University, England, July 16-20, 1990, Proceedings*, volume 443 of *Lecture Notes in Computer Science*, pages 322–335. Springer.
6. Alur, R. and Dill, D. L. (1994). A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235.
7. Alur, R. and Madhusudan, P. (2004). Decision problems for timed automata: A survey. In Bernardo, M. and Corradini, F., editors, *Formal Methods for the Design of Real-Time Systems, International School on Formal Methods for the Design of Computer, Communication and Software Systems, SFM-RT 2004, Bertinoro, Italy, September 13-18, 2004, Revised Lectures*, volume 3185 of *Lecture Notes in Computer Science*, pages 1–24. Springer.
8. Arnold, V. I. (1989). *Mathematical methods of classical mechanics*, volume 60 of *Graduate Texts in Mathematics*. Springer, second edition.
9. Artobolevskii, I. (1964). *Mechanisms for the Generation of Plane Curves*. Macmillan, New York. Translated by R.D. Wills and W. Johnson.
10. Asarin (2004). Challenges in timed languages: From applied theory to basic theory. *Bulletin of the European Association for Theoretical Computer Science*, 83:106–120.
11. Asarin, E. (1995). Chaos and undecidabilty (draft). Avalaible in `http://www.liafa.jussieu.fr/$\tilde{\}$asarin/`.

12. Asarin, E. (1998). Equations on timed languages. In Henzinger, T. A. and Sastry, S., editors, *Hybrid Systems: Computation and Control, First International Workshop, HSCC'98, Berkeley, CA, April 13-15, 1998, Proceedings*, volume 1386 of *Lecture Notes in Computer Science*, pages 1–12. Springer.
13. Asarin, E. (2006). Noise and decidability. Continuous Dynamics and Computability Colloquium. Video and sound available trough "Diffusion des savoirs de l'Ecole Normale Supérieure," on `http://www.diffusion.ens.fr/en/index.php?res=conf\&idconf=1226`.
14. Asarin, E. and Bouajjani, A. (2001). Perturbed Turing machines and hybrid systems. In *Proceedings of the 16th Annual IEEE Symposium on Logic in Computer Science*, pages 269–278, Los Alamitos, CA. IEEE Computer Society Press.
15. Asarin, E., Caspi, P., and Maler, O. (1997). A Kleene theorem for timed automata. In *Proceedings, 12th Annual IEEE Symposium on Logic in Computer Science*, pages 160–171, Warsaw, Poland. IEEE Computer Society Press.
16. Asarin, E., Caspi, P., and Maler, O. (2002). Timed regular expressions. *Journal of the ACM*, 49(2):172–206.
17. Asarin, E. and Collins, P. (2005). Noisy Turing machines. In Caires, L., Italiano, G. F., Monteiro, L., Palamidessi, C., and Yung, M., editors, *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings*, volume 3580 of *Lecture Notes in Computer Science*, pages 1031–1042. Springer.
18. Asarin, E. and Dima, C. (2002). Balanced timed regular expressions. *Electronic Notes in Theoretical Computer Science*, 68(5).
19. Asarin, E. and Maler, O. (1998). Achilles and the tortoise climbing up the arithmetical hierarchy. *Journal of Computer and System Sciences*, 57(3):389–398.
20. Asarin, E., Maler, O., and Pnueli, A. (1995). Reachability analysis of dynamical systems having piecewise-constant derivatives. *Theoretical Computer Science*, 138(1):35–65.
21. Asarin, E. and Schneider, G. (2002). Widening the boundary between decidable and undecidable hybrid systems. In Brim, L., Jancar, P., Kretínský, M., and Kucera, A., editors, *CONCUR 2002 - Concurrency Theory, 13th International Conference, Brno, Czech Republic, August 20-23, 2002, Proceedings*, volume 2421 of *Lecture Notes in Computer Science*, pages 193–208. Springer.
22. Asarin, E., Schneider, G., and Yovine, S. (2001). On the decidability of the reachability problem for planar differential inclusions. In Benedetto, M. D. D. and Sangiovanni-Vincentelli, A. L., editors, *Hybrid Systems: Computation and Control, 4th International Workshop, HSCC 2001, Rome, Italy, March 28-30, 2001, Proceedings*, volume 2034 of *Lecture Notes in Computer Science*, pages 89–104. Springer.
23. Beauquier, D. (1998). Pumping lemmas for timed automata. In Nivat, M., editor, *Foundations of Software Science and Computation Structure, First International Conference, FoSSaCS'98, Held as Part of the European Joint Conferences on the Theory and Practice of Software, ETAPS'98, Lisbon, Portugal, March 28 - April 4, 1998, Proceedings*, volume 1378 of *Lecture Notes in Computer Science*, pages 81–94. Springer.
24. Ben-Hur, A., Feinberg, J., Fishman, S., and Siegelmann, H. T. (2003). Probabilistic analysis of a differential equation for linear programming. *Journal of Complexity*, 19(4):474–510.
25. Ben-Hur, A., Feinberg, J., Fishman, S., and Siegelmann, H. T. (2004a). Random matrix theory for the analysis of the performance of an analog computer: a scaling theory. *Physics Letters A*, 323(3–4):204–209.
26. Ben-Hur, A., Roitershtein, A., and Siegelmann, H. T. (2004b). On probabilistic analog automata. *Theoretical Computer Science*, 320(2–3):449–464.

27. Ben-Hur, A., Siegelmann, H. T., and Fishman, S. (2002). A theory of complexity for continuous time systems. *Journal of Complexity*, 18(1):51–86.
28. Blondel, V. D. and Tsitsiklis, J. N. (1999). Complexity of stability and controllability of elementary hybrid systems. *Automatica*, 35(3):479–489.
29. Blondel, V. D. and Tsitsiklis, J. N. (2000). A survey of computational complexity results in systems and control. *Automatica*, 36(9):1249–1274.
30. Blum, L., Cucker, F., Shub, M., and Smale, S. (1998). *Complexity and Real Computation*. Springer.
31. Blum, L., Shub, M., and Smale, S. (1989). On a theory of computation and complexity over the real numbers; NP completeness, recursive functions and universal machines. *Bulletin of the American Mathematical Society*, 21(1):1–46.
32. Bournez, O. (1999a). Achilles and the Tortoise climbing up the hyper-arithmetical hierarchy. *Theoretical Computer Science*, 210(1):21–71.
33. Bournez, O. (1999b). *Complexité Algorithmique des Systèmes Dynamiques Continus et Hybrides*. PhD thesis, Ecole Normale Supérieure de Lyon.
34. Bournez, O. (2006). How much can analog and hybrid systems be proved (super-)Turing. *Applied Mathematics and Computation*, 178(1):58–71.
35. Bournez, O., Campagnolo, M. L., Graça, D. S., and Hainry, E. (2007). Polynomial differential equations compute all real computable functions on computable compact intervals. *Journal of Complexity*. To appear.
36. Bournez, O. and Hainry, E. (2005). Elementarily computable functions over the real numbers and $\mathbb{R}$-sub-recursive functions. *Theoretical Computer Science*, 348(2–3): 130–147.
37. Bournez, O. and Hainry, E. (2006). Recursive analysis characterized as a class of real recursive functions. *Fundamenta Informaticae*, 74(4):409–433.
38. Bouyer, P., Dufourd, C., Fleury, E., and Petit, A. (2000a). Are timed automata updatable? In Emerson, E. A. and Sistla, A. P., editors, *Computer Aided Verification, 12th International Conference, CAV 2000, Chicago, IL, July 15-19, 2000, Proceedings*, volume 1855 of *Lecture Notes in Computer Science*, pages 464–479. Springer.
39. Bouyer, P., Dufourd, C., Fleury, E., and Petit, A. (2000b). Expressiveness of updatable timed automata. In Nielsen, M. and Rovan, B., editors, *Mathematical Foundations of Computer Science 2000, 25th International Symposium, MFCS 2000, Bratislava, Slovakia, August 28 - September 1, 2000, Proceedings*, volume 1893 of *Lecture Notes in Computer Science*, pages 232–242. Springer.
40. Bouyer, P. and Petit, A. (1999). Decomposition and composition of timed automata. In Wiedermann, J., van Emde Boas, P., and Nielsen, M., editors, *Automata, Languages and Programming, 26th International Colloquium, ICALP'99, Prague, Czech Republic, July 11-15, 1999, Proceedings*, volume 1644 of *Lecture Notes in Computer Science*, pages 210–219. Springer.
41. Bouyer, P. and Petit, A. (2002). A Kleene/Büchi-like theorem for clock languages. *Journal of Automata, Languages and Combinatorics*, 7(2):167–186.
42. Bowles, M. D. (1996). U.S. technological enthusiasm and British technological skepticism in the age of the analog brain. *IEEE Annals of the History of Computing*, 18(4): 5–15.
43. Branicky, M. S. (1995a). *Studies in Hybrid Systems: Modeling, Analysis, and Control*. PhD thesis, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA.
44. Branicky, M. S. (1995b). Universal computation and other capabilities of hybrid and continuous dynamical systems. *Theoretical Computer Science*, 138(1):67–100.

45. Brihaye, T. (2006). A note on the undecidability of the reachability problem for o-minimal dynamical systems. *Math. Log. Q*, 52(2):165–170.
46. Brihaye, Th. and Michaux, Ch. (2005). On the expressiveness and decidability of o-minimal hybrid systems. *Journal of Complexity*, 21(4):447–478.
47. Brockett, R. W. (1989). Smooth dynamical systems which realize arithmetical and logical operations. In Nijmeijer, H. and Schumacher, J. M., editors, *Three Decades of Mathematical Systems Theory*, volume 135 of *Lecture Notes in Computer Science*, pages 19–30. Springer.
48. Brockett, R. W. (1991). Dynamical systems that sort lists, diagonalize matrices, and solve linear programming problems. *Linear Algebra and its Applications*, 146:79–91.
49. Brockett, R. W. (1994). Dynamical systems and their associated automata. In U. Helmke, R. M. and Saurer, J., editors, *Systems and Networks: Mathematical Theory and Applications*, volume 77, pages 49–69. Akademi-Verlag, Berlin.
50. Bush, V. (1931). The differential analyser. *Journal of the Franklin Institute*, 212(4): 447–488.
51. Calude, C. S. and Pavlov, B. (2002). Coins, Quantum measurements, and Turing's barrier. *Quantum Information Processing*, 1(1-2):107–127.
52. Campagnolo, M., Moore, C., and Costa, J. F. (2000). Iteration, inequalities, and differentiability in analog computers. *Journal of Complexity*, 16(4):642–660.
53. Campagnolo, M., Moore, C., and Costa, J. F. (2002). An analog characterization of the Grzegorczyk hierarchy. *Journal of Complexity*, 18(4):977–1000.
54. Campagnolo, M. L. (2001). *Computational complexity of real valued recursive functions and analog circuits*. PhD thesis, IST, Universidade Técnica de Lisboa.
55. Campagnolo, M. L. (2002). The complexity of real recursive functions. In Calude, C., Dinneen, M., and Peper, F., editors, *Unconventional Models of Computation, UMC'02*, Volume 2509 in *Lecture Notes in Computer Science*, pages 1–14. Springer.
56. Campagnolo, M. L. (2004). Continuous time computation with restricted integration capabilities. *Theoretical Computer Science*, 317(4):147–165.
57. Campagnolo, M. L. and Ojakian, K. (2007). The elementary computable functions over the real numbers: applying two new techniques. *Archive for Mathematical Logic*. To appear.
58. Casey, M. (1996). The dynamics of discrete-time computation, with application to recurrent neural networks and finite state machine extraction. *Neural Computation*, 8: 1135–1178.
59. Casey, M. (1998). Correction to proof that recurrent neural networks can robustly recognize only regular languages. *Neural Computation*, 10:1067–1069.
60. Ceraens, K. and Viksna, J. (1996). Deciding reachability for planar multi-polynomial systems. In *Hybrid Systems III*, volume 1066 of *Lecture Notes in Computer Science*, page 389. Springer-Verlag.
61. Church, A. (1936). An unsolvable problem of elementary number theory. *American Journal of Mathematics,*, 58:345–363. Reprinted in [73].
62. Clote, P. (1998). Computational models and function algebras. In Griffor, E. R., editor, *Handbook of Computability Theory*, pages 589–681. North-Holland, Amsterdam.
63. Coddington, E. A. and Levinson, N. (1972). *Theory of Ordinary Differentiel Equations*. McGraw-Hill.
64. Collins, P. (2005). Continuity and computability on reachable sets. *Theoretical Computer Science*, 341:162–195.
65. Collins, P. and Lygeros, J. (2005). Computability of finite-time reachable sets for hybrid systems. In *Proceedings of the 44th IEEE Conference on Decision and Control and the European Control Conference*, pages 4688–4693. IEEE Computer Society Press.

66. Collins, P. and van Schuppen, J. H. (2004). Observability of piecewise-affine hybrid systems. In Alur, R. and Pappas, G. J., editors, *Hybrid Systems: Computation and Control, 7th International Workshop, HSCC 2004, Philadelphia, PA, March 25-27, 2004, Proceedings*, volume 2993 of *Lecture Notes in Computer Science*, pages 265–279. Springer.
67. Copeland, B. J. (1998). Even Turing machines can compute uncomputable functions. In Calude, C., Casti, J., and Dinneen, M., editors, *Unconventional Models of Computations*. Springer.
68. Copeland, B. J. (2002). Accelerating Turing machines. *Minds and Machines*, 12: 281–301.
69. Costa, J. F. and Mycka, J. (2006). The conjecture $P \neq NP$ given by some analytic condition. In Bekmann, A., Berger, U., Löwe, B., and Tucker, J., editors, *Logical Approaches to Computational Barriers, Second conference on Computability in Europe, CiE 2006*, pages 47–57, Swansea, UK. Report CSR 7-26, Report Series, University of Wales Swansea Press, 2006.
70. Coward, D. (2006). Doug Coward's Analog Computer Museum. `http://dcoward.best.vwh.net/analog/`.
71. Davies, E. B. (2001). Building infinite machines. *The British Journal for the Philosophy of Science*, 52:671–682.
72. Dee, D. and Ghil, M. (1984). Boolean difference equations, I: Formulation and dynamic behavior. *SIAM Journal on Applied Mathematics*, 44(1):111–126.
73. Davis M. (ed.) (1965) *The Undecidable: Basic Papers on Undecidable Propositions, Unsolvable Problems and Computable Functions*, Raven, NY.
74. Delvenne, J.-C., Kurka, P., and Blondel, V. D. (2004). Computational universality in symbolic dynamical systems. In Margenstern, M., editor, *MCU: International Conference on Machines, Computations, and Universality*, volume 3354 of *Lecture Notes in Computer Science*, pages 104–115. Springer.
75. Deutsch, D. (1985). Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society (London), Series A*, 400:97–117.
76. Durand-Lose, J. (2005). Abstract geometrical computation: Turing-computing ability and undecidability. In Cooper, S. B., Löwe, B., and Torenvliet, L., editors, *New Computational Paradigms, First Conference on Computability in Europe, CiE 2005, Amsterdam, The Netherlands, June 8-12, 2005, Proceedings*, volume 3526 of *Lecture Notes in Computer Science*, pages 106–116. Springer.
77. Earman, J. and Norton, J. D. (1993). Forever is a day: Supertasks in Pitowksy and Malament-Hogarth spacetimes. *Philosophy of Science*, 60(1):22–42.
78. Etesi, G. and Németi, I. (2002). Non-Turing computations via Malament-Hogarth spacetimes. *International Journal Theoretical Physics*, 41:341–370.
79. Faybusovich, L. (1991a). Dynamical systems which solve optimization problems with linear constraints. *IMA Journal of Mathematical Control and Information*, 8:135–149.
80. Faybusovich, L. (1991b). Hamiltonian structure of dynamical systems which solve linear programming problems. *Physics*, D53:217–232.
81. Filippov, A. (1988). *Differential equations with discontinuous right-hand sides.* Kluwer Academic Publishers.
82. Finkel, O. (2006). On the shuffle of regular timed languages. *Bulletin of the European Association for Theoretical Computer Science*, 88:182–184. Technical Contributions.
83. Foy, J. (2004). A dynamical system which must be stable whose stability cannot be proved. *Theoretical Computer Science*, 328(3):355–361.
84. Francisco, A. P. L. (2002). Finite automata over continuous time. Diploma Thesis. Universidade Técnica de Lisboa, Instituto Superior Técnico.

85. Fränzle, M. (1999). Analysis of hybrid systems: An ounce of realism can save an infinity of states. In Flum, J. and Rodríguez-Artalejo, M., editors, *Computer Science Logic (CSL'99)*, volume 1683 of *Lecture Notes in Computer Science*, pages 126–140. Springer Verlag.
86. Gori, M. and Meer, K. (2002). A step towards a complexity theory for analog systems. *Mathematical Logic Quarterly*, 48(Suppl. 1):45–58.
87. Graça, D. (2002). The general purpose analog computer and recursive functions over the reals. Master's thesis, IST, Universidade Técnica de Lisboa.
88. Graça, D. S. (2004). Some recent developments on Shannon's general purpose analog computer. *Mathematical Logic Quarterly*, 50(4–5):473–485.
89. Graça, D. S. and Costa, J. F. (2003). Analog computers and recursive functions over the reals. *Journal of Complexity*, 19(5):644–664.
90. Graça, D., Campagnolo, M., and Buescu, J. (2005). Robust simulations of Turing machines with analytic maps and flows. In Cooper, B., Loewe, B., and Torenvliet, L., editors, *Proceedings of CiE'05, New Computational Paradigms*, volume 3526 of *Lecture Notes in Computer Science*, pages 169–179. Springer.
91. Graça, D. S., Campagnolo, M. L., and Buescu, J. (2007). Computability with polynomial differential equations. *Advances in Applied Mathematics*. To appear.
92. Graça, D. S., Zhong, N., and Buescu, J. (2006). Computability, noncomputability and undecidability of maximal intervals of IVPs. *Transactions of the American Mathematical Society*. To appear.
93. Grigorieff, S. and Margenstern, M. (2004). Register cellular automata in the hyperbolic plane. *Fundamenta Informaticae*, 1(61):19–27.
94. Gruska, J. (1997). *Foundations of Computing*. International Thomson Publishing.
95. Gupta, V., A., T., and Jagadeesan, R. (1997). Robust timed automata. In Maler, O., editor, *Hybrid and Real-Time Systems, International Workshop. HART'97, Grenoble, France, March 26-28, 1997, Proceedings*, volume 1201 of *Lecture Notes in Computer Science*, pages 331–345. Springer.
96. Head, T. (1987). Formal language theory and DNA: An analysis of the generative capacity of specific recombinant behaviors. *Bulletin of Mathematical Biology*, 49:737–759.
97. Helmke, U. and Moore, J. (1994). *Optimization and Dynamical Systems*. Communications and Control Engineering Series. Springer Verlag, London.
98. Henzinger, T. A., Kopke, P. W., Puri, A., and Varaiya, P. (1998). What's decidable about hybrid automata? *Journal of Computer and System Sciences*, 57(1):94–124.
99. Henzinger, T. A. and Raskin, J.-F. (2000). Robust undecidability of timed and hybrid systems. In Lynch, N. A. and Krogh, B. H., editors, *Hybrid Systems: Computation and Control, Third International Workshop, HSCC 2000, Pittsburgh, PA, March 23-25, 2000, Proceedings*, volume 1790 of *Lecture Notes in Computer Science*, pages 145–159. Springer.
100. Hirsch, M. W., Smale, S., and Devaney, R. (2003). *Differential Equations, Dynamical Systems, and an Introduction to Chaos*. Elsevier Academic Press.
101. Hogarth, M. (1994). Non-Turing computers and non-Turing computability. In *Proceedings of the Philosophy of Science Association (PSA'94)*, volume 1, pages 126–138.
102. Hogarth, M. (1996). *Predictability, Computability and Spacetime*. PhD thesis, Sidney Sussex College, Cambridge.
103. Hogarth, M. (2006). Non-Turing computers are the new non-Eucliedean geometries. In *Future Trends in Hypercomputation*. Sheffield, 11–13 September 2006. Available for download on www.hypercomputation.net.
104. Hogarth, M. L. (1992). Does general relativity allow an observer to view an eternity in a finite time? *Foundations of Physics Letters*, 5:173–181.

105. Hopfield, J. J. (1984). Neural networks with graded responses have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences of the United States of America*, 81:3088–3092.
106. Hopfield, J. J. and Tank, D. W. (1985). 'Neural' computation of decisions in optimization problems. *Biological Cybernetics*, 52:141–152.
107. Hoyrup, M. (2006). Dynamical systems: stability and simulability. Technical report, Département d'Informatique, ENS Paris.
108. Kempe, A. (1876). On a general method of describing plane curves of the n–th degree by linkwork. *Proceedings of the London Mathematical Society*, 7:213–216.
109. Kieu, T. D. (2004). Hypercomputation with quantum adiabatic processes. *Theoretical Computer Science*, 317(1-3):93–104.
110. Kleene, S. C. (1936). General recursive functions of natural numbers. *Mathematical Annals*, 112:727–742. Reprinted in [73].
111. Ko, K.-I. (1983). On the computational complexity of ordinary differential equations. *Information and Control*, 58(1-3):157–194.
112. Ko, K.-I. (1991). *Complexity Theory of Real Functions*. Progress in Theoretical Computer Science. Birkhäuser, Boston.
113. Koiran, P. (2001). The topological entropy of iterated piecewise affine maps is uncomputable. *Discrete Mathematics & Theoretical Computer Science*, 4(2):351–356.
114. Koiran, P., Cosnard, M., and Garzon, M. (1994). Computability with low-dimensional dynamical systems. *Theoretical Computer Science*, 132(1-2):113–128.
115. Koiran, P. and Moore, C. (1999). Closed-form analytic maps in one and two dimensions can simulate universal Turing machines. *Theoretical Computer Science*, 210(1):217–223.
116. Korovina, M. V. and Vorobjov, N. (2004). Pfaffian hybrid systems. In Marcinkowski, J. and Tarlecki, A., editors, *Computer Science Logic, 18th International Workshop, CSL 2004, 13th Annual Conference of the EACSL, Karpacz, Poland, September 20-24, 2004, Proceedings*, volume 3210 of *Lecture Notes in Computer Science*, pages 430–441. Springer.
117. Korovina, M. V. and Vorobjov, N. (2006). Upper and lower bounds on sizes of finite bisimulations of Pfaffian hybrid systems. In Beckmann, A., Berger, U., Löwe, B., and Tucker, J. V., editors, *Logical Approaches to Computational Barriers, Second Conference on Computability in Europe, CiE 2006, Swansea, UK, June 30-July 5, 2006, Proceedings*, volume 3988 of *Lecture Notes in Computer Science*, pages 267–276. Springer.
118. Kurganskyy, O. and Potapov, I. (2005). Computation in one-dimensional piecewise maps and planar pseudo-billiard systems. In Calude, C., Dinneen, M. J., Paun, G., Pérez-Jiménez, M. J., and Rozenberg, G., editors, *Unconventional Computation, 4th International Conference, UC 2005, Sevilla, Spain, October 3-7, 2005, Proceedings*, volume 3699 of *Lecture Notes in Computer Science*, pages 169–175. Springer.
119. Lafferriere, G. and Pappas, G. J. (2000). O-minimal hybrid systems. *Mathematics of Control, Signals, and Systems*, 13:1–21.
120. Legenstein, R. and Maass, W. (2007). What makes a dynamical system computationally powerful? In Haykin, S., Principe, J. C., Sejnowski, T., and McWhirter, J., editors, *New Directions in Statistical Signal Processing: From Systems to Brain*, pages 127–154. MIT Press, Cambridge, MA.
121. Lipshitz, L. and Rubel, L. A. (1987). A differentially algebraic replacement theorem, and analog computability. *Proceedings of the American Mathematical Society*, 99(2): 367–372.
122. Lipton, R. J. (1995). DNA solution of hard computational problems. *Science*, 268: 542–545.

123. Loff, B. (2007). A functional characterisation of the analytical hierarchy. In *Computability in Europe 2007: Computation and Logic in the Real World.*
124. Loff, B., Costa, J. F., and Mycka, J. (2007a). Computability on reals, infinite limits and differential equations. *Applied Mathematics and Computation*. To appear.
125. Loff, B., Costa, J. F., and Mycka, J. (2007b). The new promise of analog computation. In *Computability in Europe 2007: Computation and Logic in the Real World.*
126. Maass, W. (1996a). Lower bounds for the computational power of networks of spiking neurons. *Neural Computation*, 8(1):1–40.
127. Maass, W. (1996b). On the computational power of noisy spiking neurons. In Touretzky, D., Mozer, M. C., and Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems*, volume 8, pages 211–217. MIT Press, Cambridge, MA.
128. Maass, W. (1997a). A model for fast analog computations with noisy spiking neurons. In Bower, J., editor, *Computational Neuroscience: Trends in research*, pages 123–127.
129. Maass, W. (1997b). Networks of spiking neurons: the third generation of neural network models. *Neural Networks*, 10:1659–1671.
130. Maass, W. (1999). Computing with spiking neurons. In Maass, W. and Bishop, C. M., editors, *Pulsed Neural Networks*, pages 55–85. MIT Press, Cambridge, MA.
131. Maass, W. (2002). Computing with spikes. *Special Issue on Foundations of Information Processing of TELEMATIK*, 8(1):32–36.
132. Maass, W. (2003). Computation with spiking neurons. In Arbib, M. A., editor, *The Handbook of Brain Theory and Neural Networks*, pages 1080–1083. MIT Press, Cambridge, MA. 2nd edition.
133. Maass, W. and Bishop, C. (1998). *Pulsed Neural Networks*. MIT Press, Cambridge, MA.
134. Maass, W., Joshi, P., and Sontag, E. D. (2007). Computational aspects of feedback in neural circuits. *Public Library of Science Computational Biology*, 3(1):1–20. e165.
135. Maass, W. and Natschläger, T. (2000). A model for fast analog computation based on unreliable synapses. *Neural Computation*, 12(7):1679–1704.
136. Maass, W. and Orponen, P. (1998). On the effect of analog noise in discrete-time analog computations. *Neural Computation*, 10(5):1071–1095.
137. Maass, W. and Ruf, B. (1999). On computation with pulses. *Information and Computation*, 148(2):202–218.
138. Maass, W. and Sontag, E. (1999). Analog neural nets with gaussian or other common noise distributions cannot recognize arbitrary regular languages. *Neural Computation*, 11(3):771–782.
139. MacLennan, B. J. (2001). Can differential equations compute? `citeseer.ist.psu.edu/maclennan01can.html`.
140. Mills, J. (1995). Programmable VLSI extended analog computer for cyclotron beam control. Technical Report 441, Indiana University Computer Science.
141. Mills, J. W., Himebaugh, B., Allred, A., Bulwinkle, D., Deckard, N., Gopalakrishnan, N., Miller, J., Miller, T., Nagai, K., Nakamura, J., Ololoweye, B., Vlas, R., Whitener, P., Ye, M., , and Zhang, C. (2005). Extended analog computers: A unifying paradigm for VLSI, plastic and colloidal computing systems. In *Workshop on Unique Chips and Systems (UCAS-1). Held in conjunction with IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS05)*, Austin, Texas.
142. Müller, N. and Moiske, B. (1993). Solving initial value problems in polynomial time. In *Proc. 22 JAIIO - PANEL '93, Part 2*, pages 283–293.
143. Moore, C. (1990). Unpredictability and undecidability in dynamical systems. *Physical Review Letters*, 64(20):2354–2357.

144. Moore, C. (1991). Generalized shifts: unpredictability and undecidability in dynamical systems. *Nonlinearity*, 4(3):199–230.
145. Moore, C. (1996). Recursion theory on the reals and continuous-time computation. *Theoretical Computer Science*, 162(1):23–44.
146. Moore, C. (1998a). Dynamical recognizers: real-time language recognition by analog computers. *Theoretical Computer Science*, 201(1–2):99–136.
147. Moore, C. (1998b). Finite-dimensional analog computers: Flows, maps, and recurrent neural networks. In Calude, C. S., Casti, J. L., and Dinneen, M. J., editors, *Unconventional Models of Computation (UMC'98)*. Springer.
148. Murray, J. D. (2002). *Mathematical Biology. I: An Introduction*. Springer, third edition.
149. Mycka, J. and Costa, J. F. (2004). Real recursive functions and their hierarchy. *Journal of Complexity*, 20(6):835–857.
150. Mycka, J. and Costa, J. F. (2005). What lies beyond the mountains? Computational systems beyond the Turing limit. *European Association for Theoretical Computer Science Bulletin*, 85:181–189.
151. Mycka, J. and Costa, J. F. (2006). The $P \neq NP$ conjecture in the context of real and complex analysis. *Journal of Complexity*, 22(2):287–303.
152. Mycka, J. and Costa, J. F. (2007). A new conceptual framework for analog computation. *Theoretical Computer Science*, 374:277–290.
153. Natschläger, T. and Maass, W. (2002). Spiking neurons and the induction of finite state machines. *Theoretical Computer Science: Special Issue on Natural Computing*, 287(1):251–265.
154. Németi, I. and Andréka, H. (2006). New physics and hypercomputation. In Wiedermann, J., Tel, G., Pokorný, J., Bieliková, M., and Stuller, J., editors, *SOFSEM 2006: Theory and Practice of Computer Science, 32nd Conference on Current Trends in Theory and Practice of Computer Science, Merín, Czech Republic, January 21-27, 2006, Proceedings*, volume 3831 of *Lecture Notes in Computer Science*, page 63. Springer.
155. Németi, I. and Dávid, G. (2006). Relativistic computers and the Turing barrier. *Applied Mathematics and Computation*, 178:118–142.
156. Nicollin, X., Olivero, A., Sifakis, J., and Yovine, S. (1993). An approach to the description and analysis of hybrid systems. In Grossman, R. L., Nerode, A., Ravn, A. P., and Rischel, H., editors, *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 149–178. Springer.
157. Omohundro, S. (1984). Modelling cellular automata with partial differential equations. *Physica D*, 10D(1–2):128–134.
158. Orponen, P. (1994). Computational complexity of neural networks: a survey. *Nordic Journal of Computing*, 1(1):94–110.
159. Orponen, P. (1996). The computational power of discrete Hopfield nets with hidden units. *Neural Computation*, 8(2):403–415.
160. Orponen, P. (1997). A survey of continuous-time computation theory. In Du, D.-Z. and Ko, K.-I., editors, *Advances in Algorithms, Languages, and Complexity*, pages 209–224. Kluwer Academic Publishers.
161. Orponen, P. and Šíma, J. (2000). A continuous-time Hopfield net simulation of discrete neural networks. In *Proceedings of the 2nd International ICSC Symposium on Neural Computations (NC'2000)*, pages 36–42, Berlin, Germany. ICSC Academic Press, Wetaskiwin (Canada)
162. Papadimitriou, C. (2001). Algorithms, games, and the Internet. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing: Hersonissos, Crete, Greece, July 6–8, 2001*, pages 749–753, New York, NY. ACM Press.

163. Păun, G. (2002). *Membrane Computing. An Introduction*. Springer-Verlag, Berlin.
164. Post, E. (1946). A variant of a recursively unsolvable problem. *Bulletin of the American Math. Soc.*, 52:264–268.
165. Pour-El, M. and Zhong, N. (1997). The wave equation with computable initial data whose unique solution is nowhere computable. *Mathematical Logic Quarterly*, 43(4):499–509.
166. Pour-El, M. B. (1974). Abstract computability and its relation to the general purpose analog computer (some connections between logic, differential equations and analog computers). *Transactions of the American Mathematical Society*, 199:1–28.
167. Pour-El, M. B. and Richards, J. I. (1979). A computable ordinary differential equation which possesses no computable solution. *Annals of Mathematical Logic*, 17:61–90.
168. Pour-El, M. B. and Richards, J. I. (1981). The wave equation with computable initial data such that its unique solution is not computable. *Advances in Mathematics*, 39: 215–239.
169. Pour-El, M. B. and Richards, J. I. (1989). *Computability in Analysis and Physics*. Springer.
170. Puri, A. (1998). Dynamical properties of timed automata. In Ravn, A. P. and Rischel, H., editors, *Formal Techniques in Real-Time and Fault-Tolerant Systems, 5th International Symposium, FTRTFT'98, Lyngby, Denmark, September 14-18, 1998, Proceedings*, volume 1486 of *Lecture Notes in Computer Science*, pages 210–227. Springer.
171. Puri, A. and Varaiya, P. (1994). Decidability of hybrid systems with rectangular differential inclusion. In Dill, D. L., editor, *Computer Aided Verification, 6th International Conference, CAV '94, Stanford, CA. June 21-23, 1994, Proceedings*, volume 818 of *Lecture Notes in Computer Science*, pages 95–104. Springer.
172. Rabin, M. O. (1963). Probabilistic automata. *Information and Control*, 6(3):230–245.
173. Rabinovich, A. (2003). Automata over continuous time. *Theoretical Computer Science*, 300(1–3):331–363.
174. Rabinovich, A. M. and Trakhtenbrot, B. A. (1997). From finite automata toward hybrid systems (extended abstract). In Chlebus, B. S. and Czaja, L., editors, *Fundamentals of Computation Theory, 11th International Symposium, FCT '97, Kraków, Poland, September 1-3, 1997, Proceedings*, volume 1279 of *Lecture Notes in Computer Science*, pages 411–422. Springer.
175. Rubel, L. A. (1989). A survey of transcendentally transcendental functions. *American Mathematical Monthly*, 96(9):777–788.
176. Rubel, L. A. (1993). The extended analog computer. *Advances in Applied Mathematics*, 14:39–50.
177. Ruohonen, K. (1993). Undecidability of event detection for ODEs. *Journal of Information Processing and Cybernetics*, 29:101–113.
178. Ruohonen, K. (1994). Event detection for ODEs and nonrecursive hierarchies. In Karhumäki, J. and Maurer, H., editors, *Proceedings of the Colloquium in Honor of Arto Salomaa. Results and Trends in Theoretical Computer Science (Graz, Austria, June 10-11, 1994)*, volume 812 of *Lecture Notes in Computer Science*, pages 358–371. Springer, Berlin.
179. Ruohonen, K. (1996). An effective Cauchy-Peano existence theorem for unique solutions. *International Journal of Foundations of Computer Science*, 7(2):151–160.
180. Ruohonen, K. (1997a). Decidability and complexity of event detection problems for ODEs. *Complexity*, 2(6):41–53.
181. Ruohonen, K. (1997b). Undecidable event detection problems for ODEs of dimension one and two. *Theoretical Informatics and Applications*, 31(1):67–79.

182. Ruohonen, K. (2004). Chomskian hierarchies of families of sets of piecewise continuous functions. *Theory of Computing Systems*, 37(5):609–638.
183. Shannon, C. E. (1941). Mathematical theory of the differential analyser. *Journal of Mathematics and Physics MIT*, 20:337–354.
184. Shor, P. W. (1994). Algorithms for quantum computation: Discrete logarithms and factoring. In Goldwasser, S., editor, *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, Los Alamitos, CA. IEEE Computer Society Press.
185. Siegelmann, H. T. and Fishman, S. (1998). Analog computation with dynamical systems. *Physica D*, 120:214–235.
186. Siegelmann, H. T. and Sontag, E. D. (1994). Analog computation via neural networks. *Theoretical Computer Science*, 131(2):331–360.
187. Siegelmann, H. T. and Sontag, E. D. (1995). On the computational power of neural nets. *Journal of Computer and System Sciences*, 50(1):132–150.
188. Šíma and Orponen (2003a). Exponential transients in continuous-time Liapunov systems. *Theoretical Computer Science*, 306(1–3):353–372.
189. Šíma, J. and Orponen, P. (2003b). Continuous-time symmetric Hopfield nets are computationally universal. *Neural Computation*, 15(3):693–733.
190. Šíma, J. and Orponen, P. (2003c). General-purpose computation with neural networks: A survey of complexity theoretic results. *Neural Computation*, 15(12):2727–2778.
191. Smith, W. D. (1998). Plane mechanisms and the downhill principle. `http://citeseer.ist.psu.edu/475350.html`.
192. Smith, W. D. (2006). Church's thesis meets the N-body problem. *Applied Mathematics and Computation*, 178(1):154–183.
193. Stoll, H. M. and Lee, L. S. (1988). A continuous-time optical neural network. In *IEEE Second International Conference on Neural Networks (2nd ICNN'88)*, volume II, pages 373–384, San Diego, CA. IEEE Society Press.
194. Svoboda, A. (1948). *Computing Mechanisms and Linkages*. McGraw Hill. Reprinted by Dover Publications in 1965.
195. Thomson, W. (1876). On an instrument for calculating the integral of the product of two given functions. In *Proceedings of the Royal Society of London*, volume 24, pages 266–276.
196. Trakhtenbrot, B. (1995). Origins and metamorphoses of the trinity: Logic, nets, automata. In Kozen, D., editor, *Proceedings of the 10th Annual IEEE Symposium on Logic in Computer Science San Diego, CA, June 26-29, 1995*, pages 506–507. IEEE Computer Society, Press.
197. Trakhtenbrot, B. A. (1999). Automata and their interaction: Definitional suggestions. In Ciobanu, G. and Paun, G., editors, *Fundamentals of Computation Theory, 12th International Symposium, FCT '99, Iasi, Romania, August 30 - September 3, 1999, Proceedings*, volume 1684 of *Lecture Notes in Computer Science*, pages 54–89. Springer.
198. Tucker, J. V. and Zucker, J. I. (2007). Computability of analog networks. *Theoretical Computer Science*, 371(1-2):115–146.
199. Turing, A. (1936). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42(2):230–265. Reprinted in [73].
200. Vergis, A., Steiglitz, K., and Dickinson, B. (1986). The complexity of analog computation. *Mathematics and Computers in Simulation*, 28(2):91–113.
201. Weihrauch, K. (2000). *Computable Analysis*. Springer.

202. Weihrauch, K. and Zhong, N. (2002). Is wave propagation computable or can wave computers beat the Turing machine? *Proceedings of the London Mathematical Society*, 85(3):312–332.
203. Welch, P. D. (2006). The extent of computation in Malament-Hogarth spacetimes. http://www.citebase.org/abstract?id=oai:arXiv.org:gr-qc/0609035.
204. Williams, M. R. (1996). About this issue. *IEEE Annals of the History of Computing*, 18(4).
205. Woods, D. and Naughton, T. J. (2005). An optical model of computation. *Theoretical Computer Science*, 334(1-3):227–258.