

A Survey on Analog Models of Computation

Olivier Bournez and Amaury Pouly

Abstract We present a survey on analog models of computations. Analog can be understood both as computing by analogy, or as working on the continuum. We consider both approaches, often intertwined, with a point of view mostly oriented by computation theory.

1 Introduction

There is a clear ambiguity about the sense of the word *analog* when talking about analog computations. Nowadays, it is often understood as being the opposite of *digital*: the former is working on continuous quantities, while the latter is working over discrete values, typically bits or words. However, historically, analog computation got its name from computation by analogy, i.e. by systems built in such a way that they evolve exactly as the system intended to model or simulate [243, 328]. These two understandings are orthogonal and various machines analog in both or one and not the other sense have been conceived [328]. Notice also that even *discrete* vs *continuous* is not a clear dichotomy, and most of the analog machines that were historically built were actually hybrid [328].

We will mainly focus on models of computation, with a point of view possibly oriented by computation theory (computability, complexity, models of computation).

All considered models can be described as particular dynamical systems: a dynamical system is mathematically defined as the action of a subgroup \mathcal{T} of \mathbb{R} on a space X , i.e. by a function (a *flow*) $\phi : \mathcal{T} \times X \rightarrow X$ satisfying the following two

Olivier Bournez

Olivier Bournez, Campus de L'Ecole Polytechnique, 91128 Palaiseau Cedex France, e-mail: bournez@lix.polytechnique.fr

Amaury Pouly

MPI-SWS, E1 5, Campus, 66123 Saarbrücken, Germany e-mail: pamaury@mpi-sws.org

equations:

$$\phi(0, x) = x \quad (1)$$

$$\phi(t, \phi(s, x)) = \phi(t + s, x). \quad (2)$$

Function $\phi(t, x)$ is intended to give the position at time t of the system if started at position x at time 0, and the above equations simply express expected properties.

Subgroups \mathcal{T} of \mathbb{R} are known to be either dense in \mathbb{R} or isomorphic to the integers. In the first case, the time is said to be *continuous*, in the latter case, *discrete*.

A dynamical system is often alternatively described by some space X and some function $f : X \rightarrow X$. Indeed, in the discrete time case, giving ϕ is equivalent to giving $f : X \rightarrow X$ with $f(x) = \phi(1, x)$: the trajectory starting from some state x_0 then corresponds to the iterations of f on x_0 from Equation (2). In the continuous time case, not all dynamical systems correspond to differential equations, but at soon as ϕ is continuously differentiable, a case covering a very wide class of systems in practice, we can write $y' = f(y)$ where $f(y) = \frac{d}{dt}\phi(t, y)|_{t=0}$. Giving ϕ is then also equivalent to giving a function $f : X \rightarrow X$: the trajectory starting from some state x_0 corresponds to the solution of the Initial Value Problem (IVP) $y' = f(y), y(0) = x_0$.

We can then classify models according to their space X : we will mainly focus in this chapter on the case where the space X involves real numbers, i.e. is *continuous*. Typically $X = \mathbb{R}^n$ or $X = \mathbb{R}^n \times \mathbb{N}^k$ or can be encoded naturally in similar spaces. We will say that such a space is *continuous*, by opposition to spaces like \mathbb{N}^k (or the set of words over a given alphabet, or the set of configurations of a model such as a Turing machine) that would correspond to a *discrete* space. Discrete time and space corresponds to classical computability and complexity and are not intended to be covered by this chapter. We will nevertheless consider some unconventional discrete time and space models, such as population protocols or chemical reaction networks, as they are unconventional and turn out to be closely related to analog models as shown by various recent results.

A classification of some of the models considered in this chapter according to this discrete/continuous time/space dichotomies is provided by Figure 1. This classification is not perfect and debatable. For example, cellular automata (that we will consider later on as a spatial model) are here considered as evolving over a discrete space. But there may also be considered as evolving over $\{0, 1\}^{\mathbb{N}}$ (i.e. Baire's space), and we agree that there is no fundamental difference between \mathbb{R} and $\{0, 1\}^{\mathbb{N}}$. As another example, the discrete-space / continuous-time quadrant is rather empty, but we agree that many models have an underlying semantics that can be expressed in the language of continuous-time Markov chains, and for example stochastic chemical reaction networks could also fall in this quadrant. Actually, many models come indeed in various flavours and could be turned into various quadrants: For example, asynchronous cellular automata can arguably be considered as being continuous time and discrete space, or continuous time and continuous space.

Notice that many quantum models of computation can also be considered possibly as analog and would fit to this description: as we have to make choices, we decided that they are out of the scope of the current chapter. Notice that analog models, in particular models in the continuum, can be seen as fitting in some way

Fig. 1: A tentative classification of some computational models, according to their space and time.

Since most of the other chapters of this book are dedicated to the computable analysis framework, we will intentionally not discuss this approach in the current chapter.

Previous surveys on the field of analog computation include Pekka Orponen’s [269], Olivier Bournez and Manuel L. Campagnolo’s [65] focusing on continuous time computation, and Bruce J. MacLennan’s chapter on “Analog Computation” of the *Encyclopedia of complexity and systems science*. [243]. A recent very instructive book about the history of analog machines has recently been published [328]. There

is also an extensive literature, mostly forgotten today, about analog machines dating to the times where most of machine programming was analog.

2 Various Analog Machines and Models

2.1 Historical Accounts

As we said, historically, analog computation was mainly referring to computation by analogy. Very instructive quotations supporting this claim can be found in [328] and [243].

Actually, several of the historical first computers presented as among the first ever built digital computers turn out to be analog in this above primary sense. This includes the ENIAC, which interestingly stands for “Electronic Numerical Integrator and Computer”. The ENIAC is often said to be “*programmed*” but the term “*wired*” would be closer to reality, since this system mainly consisted of a large collection of arithmetic machines.

It may also help to understand that the term *computer* was historically referring to a person who carried out calculations or computations. From the middle of the 20th century only, the word started to refer to a *machine* that carries out computations. Determining which systems can actually be considered as computers is a very intriguing question, related to deep philosophical questions out of the scope of the present chapter.

We however list here some of the first ever built machines that can be classified as analog: This includes Blaise Pascal’s 1642 *Pascaline*, as well as Johann Martin Hermann’s 1814 *Planimeter* (a simple device to compute surfaces based on Green’s theorem), or Bill Phillips’s 1949 *MONIAC* (*Monetary National Income Analogue Computer*, a machine that was using fluidic logic to model the behavior of an economy). The *Antikythera mechanism*, discovered close to the greek island of Antikythera in 1901, dated from earlier than 87 BC, whose purpose was to predict astronomical positions and phenomena, is also often considered as an analog computer. However, even if these machines were clearly computing various quantities or data and may or may not be called computers, we admit that even the question of whether such machines can be classified as *analog* is debatable. Indeed, many articles or books consider them as such nowadays (see e.g. [328]), but these statements are conflicting with the literature in the history of computing from historians.

However, with no contest, the first truly programmable (“*general purpose*” using Shannon’s 1941 terminology) analog computer is Vannevar’s Bush 1931 *Differential Analyser*, which is the topic of the next section due to its historical and fundamental importance.

Further detailed historical accounts about analog machines and computations can be found in the recent monograph [328]. Its author, Bernd Ulmann also maintains an informative web site with instructive pictures and videos [1].

The history of analog computation and devices, is also discussed by historians. For general literature on history of analog computers, refer to [110], [252] or [335]. For references related to history of analog instruments, see [170], [169]. The history of Differential Analyser is discussed in [271]. See also [200] and [199] for national accounts about the developments of differential analysers. Historical accounts for ENIAC can be found in [197]. Recently published accounts also include monograph [292], the analysis of the work of Douglas R. Hartree [159], Charles Babbage [156] or of particular devices or techniques [157, 158] before differential analysers.

2.2 Differential Analysers

The probably best known universal continuous time machine is the *Differential Analyzer (DA)*, built for the first time in 1931 at the MIT under the supervision of Vannevar Bush [91]. The idea of assembling integrator devices to solve differential equations dates back to Lord Kelvin in 1876 [323]. Kelvin was looking for a faster way to compute the harmonics of a function using its Fourier transform, with applications to tidal and meteorological observations. He came up with the idea of using a rotating disc-cylinder-globe system to compute the integral of a product: this is essentially the fundamental operation of the Differential Analyzer, although it took over 50 years to solve the mechanical problems involved in this machine.

The first DAs were entirely mechanical, and later became electronic: see [81, 328]. Their primary purpose was to solve differential equations, especially the ones coming from problems in engineering. By the 1960s, differential analysers were progressively discarded in favor of digital technology. Many accounts on the history and applications of these machines can be found in [328].

2.2.1 Mathematical Model of the Differential Analyzer: the GPAC

The General Purpose Analog Computer (GPAC) was introduced by Shannon in 1941 as a mathematical idealization of the Differential Analyzer [305], while he was working on this machine at the MIT to get money for his studies. A GPAC is basically a circuit made up of a finite number of units, described in Figure 2, that are interconnected by wires, possibly with loops (retroactions) on some of the wires. A function $f : \mathbb{R} \rightarrow \mathbb{R}$ is said to be *generated* if it corresponds to the value read on some of the wires of the circuit as a function of time.

The model was later refined in a series of papers [286, 229, 188, 186]. Indeed, a GPAC circuit may not define a unique function (it could have no or several solutions) which is problematic. An arguably more modern presentation of the GPAC is to use ordinary differential equations instead of circuits. The equivalence of GPAC circuits with polynomial ordinary differential equations will be discussed in Section 7.1.

Rubel also introduced the Extended Analog Computer (EAC) as an extension of the GPAC [295]. The EAC features a broader class of operations such as par-

tial differentiation and restricted limits, allowing to solve boundary value problems. Rubel’s motivation for introducing the EAC was that the GPAC was “a limited machine”. This claim has been somewhat changed since then (see Section 7.1) and the EAC has seen few theoretical developments. Although Rubel envisioned the EAC as a “purely conceptual machine”, there is ongoing research to implement it, with some currently working prototypes [254, 256, 255].

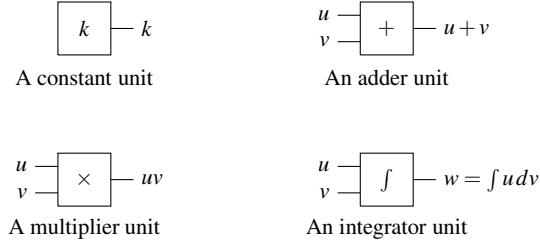


Fig. 2: Basic units used in a GPAC circuit.

2.3 Neural Networks and Deep Learning Models

In the 80’s and 90’s, *artificial neural networks* gave birth to a renewal of interest in analog computations. This enthusiasm declined until recently and the success of *deep learning* in several impressive applications in various fields of artificial intelligence, such as speech recognition, image recognition and natural language processing. The first machine able to beat all best professional *Go* players is based on deep learning technology.

Current deep learning models may have a rather complex architecture built from various modules, but most of these modules are essentially artificial neural network models. An artificial neural network (ANN) consists of many simple, connected processors called *neurons*. The state of each neuron is given by some real number called its activation value. Designated *input neurons* get their activation values from the environment. All other neurons evolve by applying a composition of a certain one-variable function σ (usually a sigmoid) with an affine combination of the activations of the neurons to which they are connected. Finally, specific neurons, considered as *output neurons*, may trigger actions on the environment: see [55] for an overview.

Most of the work related to (deep) artificial neural networks is nowadays devoted to finding architectures and weights that make an ANN exhibit a desired behavior in a given context of application. A popular library to describe corresponding architectures is *Tensor Flow* [2]. This is basically a library to describe such architectures,

and hence can be considered in many aspects as a library for particular analog computations.

A very popular and successful method to determine suitable weights for solving a given problem is *back-propagation*, which is reinforcement learning technique based on a gradient descent method applied to an error function expected to be minimized over the learning set, see [55].

Applying any gradient descent method requires a differentiable error function with respect to all involved parameters. The very large number of applications of deep learning techniques have recently led to the emergence of various other analog models of computation. All these models have in common to be differentiable end-to-end versions of models inspired by classical computability. This includes the popular *Long Term Short Memory (LSTM)* architecture [206], or the so-called *Differentiable Neural Computers* [190], or the *Neural Turing machines* [189]. The underlying principle of these constructions is to extend an artificial neural network by coupling it to an external memory resource. This external resource can also be a stack as in the so-called *Neural Network Pushdown Automata* [320] and *Neural Stack machines* [191].

The models discussed previously all work in discrete time, but models with a continuous time dynamics have also been considered, such as *symmetric Hopfield networks*. The convergence behavior of these networks has been used in various applications such as associative memory, or combinatorial optimization problems after first applications in [212].

Models based on *spiking neurons* have been claimed to be biologically more realistic [236]. Various coding methods exist to interpret the outgoing spike train as a real value number either relying on the frequency of spikes, or the timing between spikes. This yields to various ways of encoding information and the computational power of spiking neuron models has been investigated in series of papers: see [311], [236] or [179] for surveys. Recent years have seen practical implementations [249] as well as their use to solve practical problems efficiently [149].

2.4 Models from Verification

The development of algorithms and techniques for verification or control of so-called *hybrid systems* or *cyberphysical systems* have also generated several lines of research related to analog computation. *Hybrid* and *cyberphysical systems* have in common to mix discrete evolutions, often a digitally engineered controller, with continuous dynamics that often comes from the environment or from some natural continuous variables that the controller acts upon.

There is an extensive literature on the *hybrid automata* modeling approach to determine the exact frontier between decidability and non-decidability for reachability properties, according to the type of dynamics, guards, and resets allowed. The same holds for the frontier about applicability of techniques coming from control to prove properties about these systems such as stability. Providing a complete panorama on

this literature is out of the ambition of the current chapter. Classical survey references are [204, 14]. See also [65] for discussions.

This literature is the source of many *dynamic undecidability* results to which we will come back in Section 3.3.

Hybrid systems are known to exhibit the so-called *Zeno's phenomenon*. In short, they may happen to have an unbounded number of discrete transitions in a bounded (continuous) time. In this context, it is classical to distinguish various types of Zeno's behaviour: *Chattering Zeno* vs *Genuine Zeno*, following [16]. The first type can often be eliminated and corresponds more to an artefact of the model, which can be avoided by considering an appropriate notion of solution, while the second is more problematic to detect and harder to avoid in simulation [17]. We will come back in Section 4.4 to Zeno's phenomenon in the wider perspective of space time contraction phenomena in analog systems.

Despite some promising early results in the field, open problems even for very simple classes of systems still remain. In particular, the decidability of the reachability problem for piecewise affine maps on the real line is a famous open problem, with some partial progress such as [223, 47]. Several other questions seem to be closely related to that issue [32]. The decidability of point to hyperplane reachability for discrete time linear systems, known as the Skolem problem, is famously open. The *continuous time* version of *point to hyperplane* has recently been shown to be related to conjectures from transcendental number theory [122]. Hardness of *recurrent reachability for continuous linear dynamical systems* of low dimensions has been investigated in [123].

Several recent results have also focused on the hardness of *bounded time versions* of reachability problems. The complexity of the reachability problem has been characterized for hybrid automata in [87], and for piecewise affine systems in [48] and [39]. Complexity of problems or methods from control theory have also been explicitly derived [10, 11, 8, 9].

In relation with other chapters of this handbook, we mention that the recursive analysis approach has also been explored. Computability of reachable and invariant sets in the framework of computable analysis have been investigated for continuous time systems [126] and for hybrid systems [127].

A method to approximate hybrid systems with a polynomial hybrid automaton, i.e. a Taylor approximation, has also been proposed [225], as well as interval methods and Taylor model methods for ODEs [266]. There have been attempts to provide a formal semantics to Simulink based on non-standard analysis tools [53]. More generally, providing models of systems using ordinary differential equations is an hard task in practice: See e.g. very instructive discussions in [226] about all the difficulties in modeling a simple system such as Newton's pendulum.

2.4.1 Timed Automata

Timed automata [13] can be considered as a restricted version of hybrid systems for which decidability of reachability holds. They can also be considered as an exten-

sion of finite automata with clocks. The model has clear practical applications and is at the heart of several computer tools for verification: see e.g. recent survey [78].

From a more fundamental point of view, timed automata can be seen as language recognizers [13], and there has been various attempts to generalize concepts from finite automata theory to this framework. This includes closure properties of recognized languages [13], pumping lemmas [41] as well as variants of Kleene's theorem [26, 27, 28, 79, 80, 29]. For a recent survey about timed automata, see [78].

2.5 *Blum-Shub-Smale's Model*

The Blum-Shub-Smale (BSS) model [58] has been introduced as a discrete time model of computation over the reals in order to discuss hardness of problems in algebraic complexity. In the initial presentation of the model, operations of the field \mathbb{R} are assumed to be realizable at unit cost, leading to classes such as $P_{\mathbb{R}}$ and $NP_{\mathbb{R}}$, with complete problems such as the existence of a real zero of a given polynomial. Later, it has been generalized to other fields or rings with extended or restricted operations [57], or to an abstract model over arbitrary logical structures [285]. Notice that classical discrete computability models have been generalized to abstract structures in various ways in parallel, and also before [58]: See e.g. [174], [260] or [325].

The obtained computability and complexity theory subsumes classical discrete computation theory since the latter can be seen as the specific case of logical structures with a finite domain.

There is an extensive literature on the related computation theory. Many results have been obtained in this model, mostly studying the corresponding complexity classes, and their relations with classical questions in computability theory, or providing lower or upper complexity bounds on various problems based in this framework: See [57, 285].

Recent results include non-trivial generalizations to this framework of Toda's theorem [36, 37] or the PCP theorem [35], interactive proofs [34], Ladner's result [246], as well as separation of degrees [178, 177].

The model is different in spirit from most others discussed in this section, as it is usually not considered to be an attempt at modeling analog machines in a realistic way, but rather as a mathematical tool. It has proven to be relevant for the discussion of lower bounds in algebraic complexity, or for some classical questions from complexity theory in a wider generalized setting, where sometimes complexity classes such as P and NP can be separated. The model has also clear connexions with the generalized finite automata models discussed in Section 5.3.

2.6 Natural Computing

The interest in unconventional models of computation, and in particular for natural computing, has revived interest in analog computing.

2.6.1 Dissipative Influence Dynamics

The framework of *natural algorithms*, has motivated a series of works about models of *influence systems* and their computational capabilities. A manifesto in favor of the fact that the study of natural systems can benefit from an algorithmic perspective has been published in [115]. This has motivated the exploration of several models of influence dynamics such as the Hegselmann-Krause's model. In particular, bounds on the time required by a group of birds to stabilize in a standard bird flocking model have been established in [116]. Turing completeness and almost sure asymptotic periodicity of diffusive influence systems have also been obtained in [118]. For a general discussion on the merits and challenges of an algorithmic approach to natural algorithms, see [117].

2.6.2 Physarum Computing

Physarum polycephalum is a slime mold that has been shown to be able to solve various natural problems: this includes realization of boolean logic gates, implementation of delay in computing circuits, geometry computations such as Voronoi diagrams or Delaunay triangulations, or computation of shortest paths [5]. Models of various aspects of its behaviour have been established. Convergence proofs and complexity bounds for computing shortest paths have been investigated [42], based on mathematical model for the slime's behavior proposed in the form of a coupled system of differential equations [322]. Implementation of Kolmogorov-Uspensky in biological substrate has also been investigated [4]. For a survey about these fields of research, see [5].

2.6.3 DNA and Molecular Computing

Areas such as *systems biology* aim at understanding complex biological processes in terms of their basic mechanisms at the molecular level. Many attempts of applying concepts and tools from theoretical computer science to this framework (logic, algebra, etc..) have been investigated, with numerous successes and concrete software systems: see [165].

However, even if the primary purpose of these fields was to explain concrete biological features, various approaches have considered computations by chemical reactions as a programming tool. The idea is to consider computations by reactions as programs to solve various tasks, as nature does in the context of cell biology, but

not restricting to this context. In particular, various attempts to relate this concept of programming to computation theory have been proposed. In most of these approaches, the underlying principle is to get inspiration from concrete chemical phenomenon in order to derive abstract models of computation which are potentially usable. This includes the model of biomolecular computation [198], the *Chemical Abstract Machine* [54], *Membrane Computing* models [273], *Biochemical Ground Form* process algebra's approaches [109], DNA based computation models [287], or *Chemical Reaction Networks* (CRN) discussed in Section 6.5.

Simulations of Turing machines has been demonstrated in several of the above mentioned articles, both at an abstract level or concretely. A very challenging question is to compare the actual implementations in nature of some of the tasks to other possible more efficient ones that could be derived theoretically [166].

For a presentation of natural computing models and results not covered in this chapter, see [293, 272]. In this chapter, we will only briefly discuss the case of DNA computing (below) as well as that of CRNs (Section 6.5).

DNA computing started with a work [7] which proposed to solve the directed Hamiltonian path problem on a graph using DNA as well as enzymes to implement the computation. Molecular computation had been investigated in various ways before popular Adleman's article (see e.g. [129]), including the idea of using DNA computing to implement computations of formal language theory [201]. This has been extended later on to other known NP-complete problems [230], even if the required resources have been demonstrated to be unrealistic. Approaches based on evolutionary computation to control resources have also been proposed [319]. The field developed meanwhile in many impressive ways. Visible facts include the development of a programmable molecular computing machine [235], the demonstration of the use of DNA as digital storage medium encoding a 5.27-megabit book [124], or the construction of the analog of a DNA transistor [61].

2.7 Solving Various Problems Using Dynamical Systems

Several authors have shown that certain, possibly discrete, decision or optimization problems such as graph connectivity or linear programming, can be solved by specific continuous dynamical systems. Some examples and references can be found in the papers [312, 329, 90, 168, 202, 50, 259, 56]. This has links with various mechanical computer models (e.g. "billiard ball computers") investigated by several papers in the 70s and 80s, with discussions about physical limits to computations such as thermodynamic reversibility [51, 173, 52, 244]. Relevant concepts have turned out to still be important in quantum computing (e.g. Toffoli gates, etc.). This is also related to discussions about various phenomena possibly capable of hypercomputation [245, 300]: see in particular Section 2.9 about Black Hole computations for more recent references.

Observe that, if analog computation is to be understood in the sense of computing by analogy, then almost any historical analog machine can be considered as falling

under this framework. In his monograph, Ulmann develops with sometimes great and instructive details some of the machines for particular problems such as finding zeros of a polynomial, linear algebra, optimization and simulation [328]. In more than 70 pages various historical applications in about 20 fields such as Mathematics, Physics, Mechanics, Geology and Economics are described.

The question of whether some of the discrete problems could actually be solved faster using continuous methods is very intriguing: this is the object of Section 4.4.

Many dynamical systems of the form $H' = [H, [H, N]]$, where the notation $[B, L]$ stands for $BL - LB$, have been shown to be continuously solving some particular discrete problems [90, 89] such as sorting lists, diagonalizing matrices and linear programming problems. One key property is that this equation is equivalent to some gradient flow on the space of orthogonal matrices. Many examples in this spirit are discussed in details in [203]. More recently a system of nonlinear differential equations of a similar form to sort numbers fed to the input has been investigated [183].

Notice that several discrete time algorithms or methods have some analog equivalent. This includes *Newton's method* which leads to *Newton's flow dynamics* for finding roots. *Gradient descent* methods (this includes the very popular Backpropagation method for Neural Networks) can also be considered as the (explicit) Euler's discretization method of a continuous flow, the so-called *gradient's flow*.

The use of analog methods for solving k -SAT problems has been investigated in [161]: the problem is mapped to an ordinary differential equation about which some properties are established, such as chaotic transience of trajectories above a constraint density threshold and fractality of the boundaries between the basin of attraction. The system is stated to always find solutions in polynomial continuous time, but at the expense of exponential fluctuations in its energy function [161]. An attempt to physically implement the proposed dynamic has been proposed [337]. Previous statements that k -SAT can be formulated continuously can be found in [193, 265, 330].

Notice that polynomial-size continuous-time Hopfield nets have been proven able to simulate PSPACE Turing machines [310]: This implies that even ODEs with Lyapunov-function controlled dynamics can actually do much more than solving NP-complete problems in some sense.

2.7.1 Interior Point Methods

A particular class of methods falls very naturally into this framework: the so-called *interior point methods*, which correspond to a particular class of algorithms to solve linear and nonlinear convex optimization problems. The principle, already proposed by John von Neumann, and very popular in the 1960s, is to build a continuous system whose trajectories are evolving in the interior of the feasible region. A common method to guarantee evolution in the feasible regions was the use of *barrier functions* acting as a potential energy. By making these functions tend to infinity on the boundary, the evolution is guaranteed to remain feasible.

Later, these methods were mostly considered as inefficient until Karmakar triggered a revolution in the field of optimization by providing the second polynomial time algorithm for linear programming [217, 215]. It has then been realized that Karmakar's algorithm is equivalent to a particular interior point method [180, 38]. Notice that, historically, the first polynomial time algorithm for linear programming is due to Khachiyan and is based on ellipsoid method (which is not an interior point method).

A very elegant presentation of Karmakar's algorithm and the associated flow, inspired from [24], can be found in [259], including an elegant presentation of its polynomiality based on ordinary differential equation arguments. Refer to [290] for a general introduction to interior point methods.

2.8 Distributed Computing

Recent years have also seen the birth of new classes of models in *distributed computing*. In particular, the model of *population protocols* consists of passively mobile anonymous agents, with finitely many states, that interact in pairs according to some rules, i.e. a given program [21]. *Passively mobile* means that agents have no control over the other agents with whom they will interact. The model was initially introduced in the context of sensor networks, but it is nowadays considered as a fundamental model for large passively mobile populations of agents with resource-limited anonymous mobile agents.

Most works on the model have considered these protocols as computing predicates over multisets of states. Given some input configuration, the agents have to decide whether this input satisfies a given predicate: the population of agents has to eventually *stabilize* to a configuration in which every agent is in a particular accepting (respectively rejecting) state if and only if the predicate is true (resp. false). The model is *uniform*: the program is assumed to be independent of the size of the population.

The seminal work of Angluin *et al.* [22, 21] proved that predicates decided by population protocols are precisely those on counts of agents definable by a first-order formula in Presburger arithmetic – equivalently, this corresponds to semilinear sets. An elegant proof of this result can be found in [162]. Note that this computational power is rather restricted, as multiplication for example is not expressible in Presburger arithmetic.

The model has been intensively investigated since its introduction. Several variants have been studied in order to strengthen it with additional realistic and implementable assumptions. This includes natural restrictions, like modifying hypotheses on interactions between agents (e.g., one-way communications [22], particular interaction graphs [20]). This also includes probabilistic population protocols that assumes random interactions [21]. Fault tolerance has also been considered [145], including self-stabilizing solutions [23]. For some introductory texts about population protocols, see for instance [33, 251].

The model can be seen as a particular case of the (stochastic) chemical reaction networks discussed in Section 6.5.

Here, we focus on models and results in the context of distributed computing, and mainly discuss computability issues. Covering all works devoted to variants of population protocols in the distributed computing community is out of the ambition of this chapter: see [33, 251] for surveys. Among many variants of population protocols, the *passively mobile (logarithmic space) machine model* was introduced by Chatzigiannakis *et al.* [114]. In this model, each agent carries a bounded space Turing machine, instead of a finite state automaton. In an orthogonal way, *community protocols*, where each agent has a unique identifier and can only store $O(1)$ other agent identifiers, exclusively from agents that it met, were introduced by Guerraoui and Ruppert [194]. They proved, using results about the so-called *storage modification machines* [304], that such protocols simulate Turing machines very efficiently. A hierarchy between the two models has also been studied recently, by considering the case of homonyms, that is to say when several agents may share the same identifier [67, 68].

The population protocols can also capture natural models of dynamics of some opinion spreading models by considering probabilistic rules of interactions: results on the convergence and threshold properties of the so-called Lotka-Volterra population protocol have been established [142].

Notice that many models coming from dynamics of rational agents in the context of (learning equilibria in) Game Theory can also be considered as analog distributed models of computation [331, 207].

2.8.1 Large Population Protocols

When considering probabilistic interaction rules, as in the previously mentioned settings, the underlying dynamical system is a Markov chain.

If the population of agents is large, the random process converges to its (deterministic) limit continuous dynamic given by some ordinary differential equation (also called its *mean-field limit*): this corresponds to the differential semantics discussed in Section 4.1 for chemical reactions.

These considerations led to the so-called *Large Population Protocols* [71]: real numbers which correspond to limit ratio of programmable dynamics by such model have been demonstrated to correspond precisely to algebraic numbers [71]. This results has many similarities with [213] obtained in the context of stochastic reaction networks, but with a slightly different notion of computability.

A framework to translate certain subclasses of differential equations into protocols for distributed systems has been proposed [195]. This is illustrated on several examples either taken from distributed problematics (responsibility migration, majority selection) or from classical models of populations such as Lotka-Volterra model of competition.

2.9 Black Hole Models

Black Hole computations usually refers to the study of the validity of the Physical Church Thesis (see Section 4.2) in the context of Einstein's General Relativity (GR). In other words, does GR allow for spacetimes where an observer can observe in finite time an eternity of some other device. Informally, the setup is as follows: some device (that we refer to as the computer) will try to solve some hard problem, like checking the consistency of ZFC. As soon as the computer finds a counterexample, it sends a signal to the observer, otherwise it keeps checking ZFC for eternity (since there is an infinite number of formulas to check). In parallel, an observer will manage to view the result of this infinite computation in finite time. More precisely, the observer will have a spacetime location q , at which it can check for the existence of the signal: if it receives a signal, then ZFC is inconsistent, otherwise ZFC is consistent. Hogarth proved that such spacetimes, usually referred to as *Malament-Hogarth (MH)*, can exist in theory [211]. Note that in this setting, it is crucial that q is known to the observer, so that after it reached q , the observer knows whether ZFC is consistent or not, and can use this information.

This question has received a lot of attention, especially concerning the physical realization of such a setup [160, 208, 209, 163, 210]. It has since emerged that *slowly rotating Kerr black holes*, and possibly *Reissner-Nordström (RN)* black holes, provide a plausible physical realization for these experiments. Two very good surveys on the problems and solutions to the many obstacles encountered so far have been published [267, 268]. We will mention some of these in the remaining of this section. Another important question is to characterize the extent of hypercomputation available in a MH spacetime. Hogarth originally showed that any Σ_1 set could be decided in the Kerr spacetimes. Recently, it was shown that MH (but not necessarily Kerr) spacetimes can decide all hyperarithmetic predicates on integers, but not more (under some assumptions) [333].

Without giving too many details, we now mention some aspects of the physical realization of black holes computation. An historical objection was that a *Schwarzschild black hole* (non-rotating and non-charged) has a punctual singularity to which any observer is attracted (in this setting, the observer is the one "jumping" into the black hole whereas the computer stays outside), and eventually gets crushed by the ever-growing tidal forces that tear it apart. However, both slowly rotating Kerr and RN black holes have negligible tidal forces. Furthermore, their singularity has a shape of ring that the observer can avoid forever, meaning that the observer could survive infinitely long within the black hole. In particular, astronomical evidence suggest that such Kerr black holes exist and have a size roughly that of a solar system. Another kind of objection was that this setup neglected quantum effects that do not exist in pure GR, but many of these objections have been solved, although the details depend on whether the universe is expanding or not. To summarize with a slightly exaggerated statement, the entire earth population could jump into a black hole, and continue living inside forever but now knowing whether ZFC is consistent or not. Other spacetimes, such as the *anti de Sitter*, theoretically allow to exchange the role of the observer and the computer, meaning that the observer could stay out

of the black hole and send the computer into the black hole, and get an answer in finite time.

Finally, we mention some related work on *Closed Timelike Curves* (CTCs) that, if they exist, would allow to solve PSPACE problems very quickly but do not allow for hypercomputation [3]. Another line of research is to build a logical axiomatization of spacetime, that is, building relativity theory as a theory in the sense of first-order logic. For a survey on the subject, see [18].

2.10 Spatial Models

We now consider various *spatial* models, that is to say various models which share a concept of computation based on the distribution of entities among some space and communications between these entities. We agree that our classification is debatable: some of the previous models can be already considered as such (see e.g. reaction-diffusion systems), or that some of the considered models can also be considered as continuous space (see e.g. cellular-automata that can be considered as acting over Baire's space $\{0, 1\}^{\mathbb{N}}$ homeomorphic to \mathbb{R}).

2.10.1 Computational Fields

Computational fields are analog models of spatial massive parallelism. There are motivated by the hypothesis that in various contexts it now makes sense to consider that the number of processing elements is so large that it may conveniently be considered as a continuous quantity [241, 239]. The theoretical functions of computational fields, based on tools from functional analysis, have been studied [239]. Applications of Computational fields have been explored and advocated in several articles [243, 241, 240].

2.10.2 Spatial Computing Languages

Previous computational fields models fall naturally in the more general framework of *spatial computing*: computation has become cheap enough and powerful enough that a large number of computing devices can now be embedded into many environment. As a result, a whole spectrum of *Domain Specific Languages* (DSLs) have emerged to widen the gap between the application needs of users in various domains (e.g. biology, reconfigurable computing), usually at a global level, and the programming, usually at a local level, of the increasingly complex systems of interacting computing devices. A common pattern in all those models and languages is the close relationship between the computation and the arrangement of the computing devices in space. For a survey and references, with discussions about many related aspects, see [40].

2.10.3 Cellular Automata Based Models

Cellular automata correspond to a particular class of spatial computing models. The classical model is an abstract discrete time and space model, with a rather well studied computation theory. We will focus here on developments related to analog models.

There exist several Cellular Automata (CA) inspired models, usually built as a abstraction of CA in the limit where cells are infinitely small.

One such model is based on an infinite tessellation of space-time [303]. It has been compared to classical models from computability theory and proven to be capable of hypercomputations [303].

Smooth (continuous) versions of game of life have been investigated [205], yielding very elegant dynamics. More generally, the passage from cellular automata to continuous dynamics (partial differential equations) has also been investigated in [112].

Another model with several recent developments is that of *Signal Machines* (SM) introduced in [153] where dimensionless signals are synchronously moving on a continuous space, and local update rules are used to resolve collisions. The major difference with CA is that both time and space are continuous (*i.e.* reals instead of integers). A SM configuration is given by a partial mapping from \mathbb{R} (the space) to a finite number set of *meta-signals*, essentially each signal has a type. Each SM comes with a finite set of *collision rules*. When two or more signals meet, a *collision* happens, all incoming signals are destroyed and the rule gives a list of outgoing signals that are created. Between collisions, each signal moves in a direction at a certain speed that depends on its type. Thus there are only finitely many different speeds. One can think of the speed of signals as the *slope* of its line in the 2D space-time diagram. Figure 3, illustrates this process on a simple example. In a series of papers ([154] and onwards), Durand-Lose and coauthors investigate the computation power of this model with various restrictions. It is possible to embed Black Hole computations, Turing machines, BSS and more in this model, depending on whether irrational numbers and accumulation points are allowed. Since the unrestricted model exhibit Zeno phenomenon, it is super-Turing powerful.

2.11 Other Various Models

We do not intend to cover in this document all models. Various other unconventional computers can also be classified as spatial. This includes models based on propagation and interaction of waves (reaction-diffusion computers [302, 301, 6], soliton-based computers). Or models based on propagation of pattern forms (slime mould computer [19], plant root computers, crystallisation computers). Or models based on propagation of swarms of creatures (e.g. crab gates). This also include molecular self-assembly models [164, 318].

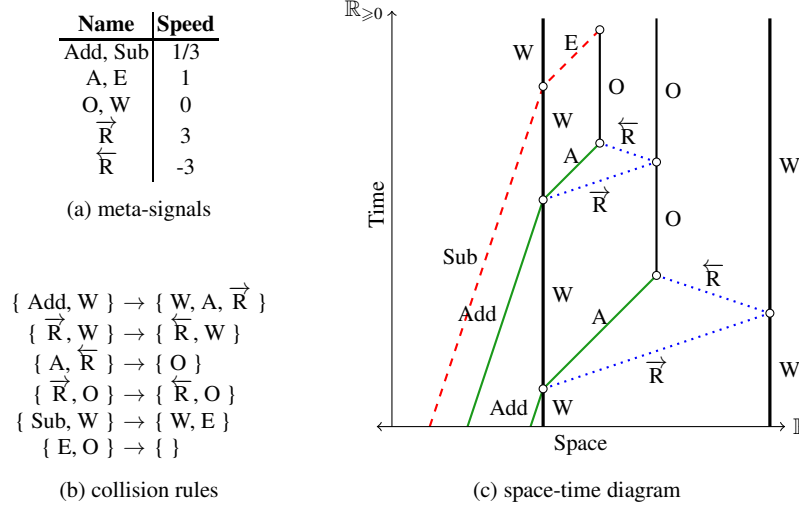


Fig. 3: Signal machine that finds the middle between two input vertical signals. The process is started by the arrival of the Add signal from the left. It triggers a sequence of collisions that results in a vertical signal O that is positioned exactly halfway between the two W signals. It is possible to generate the middle between the left W and O signal by sending another Add signal. It is also possible to suppress the leftmost O signal by sending a Sub signal from the left. Finding the middle is a key primitive in SM computations.

The frontier between analog and non-analog models of computation is not so clear, and some models are clearly at the frontier. This includes the optical models of [336], or some quantum computation models such as [148, 192, 306, 218], or finite 2-dimensional coupled map lattices that have been proven to be computationally universal [270].

Planar mechanisms were claimed to correspond to all finite algebraic curves by Kempe [216]. The initial statements of Kempe had flaws that were corrected and extended in [312]. *Tractional Motion machines* constitute an extension of this models with a rich and elegant theory discussed in [253, 155]. The fact that some functions computable in this model are not GPAC computable has been established [253]. Some discussions on limitations of machines based on the historic concept of compass and ruler constructions can be found in [261].

3 Dynamical Systems and Computations

3.1 *Arbitrary vs Rational/Computable Reals*

Before delving into the core of the topic of *dynamic undecidability*, we would like to point out a fundamental difference between discrete and continuous dynamical systems: *real numbers*. Indeed, it is clear that many objects involved in discrete computations (integers, rational numbers, graphs, etc) have a finite representation. For example, one can represent an integer in base 2 by its bits. This property is no longer true for real numbers, independently of the details of the representation. In fact, for reasonable representations (see [332] for more on the theory of representations) of the real numbers, a *single real number* can turn out to be a very powerful object. One such example is the real number, sometimes called the “*Turing number*”, whose n^{th} bit is 1 if and only if the n^{th} Turing machine halts¹. Intuitively, any powerful enough dynamical system that has access to this number can solve the Halting problem for Turing machines in finite time, and is thus super-Turing powerful. Other numbers, such that Ω of Chaitin [113], can turn out to be even more powerful. A consequence of this fact is that dynamical systems are able to solve (classically considered) uncomputable problems if one allows such numbers to appear in the system. This is why it is often important to restrict to systems with computable or rational descriptions if one does not want to consider models with some hypercomputational features.

Understanding the obtained computability theory, if this hypothesis is not done, i.e. when arbitrary (possibly non-computable) reals are allowed in the description of systems, gave birth to a whole set of developments discussed in Section 6.1 and 6.2.

In the rest of this section, in order to avoid to consider such classes of models, we assume that we restrict to rational or computable descriptions.

3.2 *Static Undecidability*

It is, however, often rather easy to get undecidability results about dynamical systems by observing that integers (or discrete sets) can be considered as particular reals, and hence that all undecidability problems known over the integers have some counterpart over the continuum [298]. This allows to map well known undecidability questions of recursive analysis (e.g. testing whether a real is zero) or of computability theory (e.g. testing whether a given polynomial with integer coefficients has an integer root) into the setting of dynamical systems. This is sometimes known as *static undecidability*. For concrete simple examples of results obtained with that spirit, see [25, 298]. More elaborate examples are the proof that determining whether a polynomial dynamical system has a Hopf bifurcation [138] or is chaotic [139] is undecidable.

¹ One needs to fix a particular enumeration of Turing machines.

3.3 Dynamic Undecidability

Dynamic undecidability, as opposed to *static undecidability* [298], corresponds to a proof technique that is often used with various models of dynamical systems. In this case, the undecidability is obtained by proving that for every Turing machine, one can build a dynamical system in the considered class able to simulate step by step the Turing machine.

To be more concrete, let us informally describe an embedding of Turing machines into dynamical systems with discrete time and continuous space. We can encode the configuration of a Turing machine using a vector (l, q, r) where $l, r \in [0, 1]$ will encode the tape (left and right side of the tape), and $q \in [0, 1]$ will encode the state, for example the i^{th} state is mapped to $\frac{i}{N}$ where N is the number of states. To encode the tape, we simply view the content of the tape as a list of digits. In the case of a binary alphabet, a word $w \in \{0, 1\}^*$ will be encoded by the real $0.w = \sum_{i=1}^{|w|} w_i 2^{-i}$. By convention, we will say that the symbol under the head is the first symbol on the right tape.

To simulate one step of the machine, we need to perform tests on the state and then apply the corresponding action. To find in which state we are, we need comparisons of the form $q = \frac{i}{N}$. To find which symbol is under the head (the first symbol of the right tape r), note that if $r_1 = 0$ then $0.r < \frac{1}{2}$, but if $r_1 = 1$ then $0.r \geq \frac{1}{2}$. Thus we only need a test of the form $r < \frac{1}{2}$. To move the head of the machine, we will only need addition and multiplication by a constant. For example, if we want to move the tape to the right, then we perform the assignment $r \leftarrow \frac{\sigma + r}{2}$ where σ is the symbol written by the head, and $l \leftarrow 2(l - \rho)$ where ρ is the first symbol of the left tape. Note that we can easily have access to ρ the same way we have access to σ and we can make a case distinction between $\rho = 0$ and $\rho = 1$. Finally we assign the new state with $q \leftarrow \frac{i'}{N}$ where i' is the new state.

To summarize, in order to simulate a Turing machine with a discrete time and continuous space machine, we need comparisons against fixed values ($=$ and $<$), assignment, addition and multiplication by a constant. Importantly, all the constants involved are rational numbers. One may be worried that exact comparisons between real values is an unrealistic assumption, there are two answers to this issue. First, analog models of computation are not necessarily concerned with realistic assumptions that come from the world of discrete computations (or from computable analysis's basic assumptions about computability): comparing two real numbers is a very natural operation in several continuous models of computations.

Second, and more importantly, even if exact tests are not possible, this simulation does not really require exact tests. Without going into too many details, notice that on the state, there is gap of size $\frac{1}{N}$ between the encoding of two states, thus an approximate test will suffice. Similarly for the tape, a classical trick is to encode a binary tape in base 4, by $0.w = \sum_{i=1}^{|w|} 2w_i 4^{-i}$. This way, if $r_1 = 0$ then $0.w \leq \frac{1}{6}$ (the worst case is $0.02222\dots$) but if $r_1 = 1$ then $r \geq \frac{1}{2}$, leaving again a constant gap between the two options.

All the above settings provides the basic ingredients for the possibility to simulate any Turing machine by a discrete time model. Similar *static undecidability* constructions have been used in various models: general dynamical systems [257, 296, 83, 298, 128], piecewise affine maps [220], sigmoidal neural nets [309], closed form analytic maps [221], which can be extended to be robust [187], and one dimensional restricted piecewise defined maps [223].

If one wants to simulate a Turing machine by a continuous time model, then several techniques may be used. One is to build a continuous time system such that the crossing of the dynamics with a given hyperplane (its *Poincaré's map*) corresponds to a discrete time system as above. Another one is to build a continuous time system such that its projections at discrete time $t = 0, 1, \dots$ (its *stroboscopic view*) is given by a discrete time system as above.

Usual such constructions rely on a suitable generalization of the idea of “*continuous clocks*” [83] for iterating functions over the integers. For simplicity, we will use a reformulation of these original equations, based on [102]. The idea is to start from the function $f : \mathbb{R} \rightarrow \mathbb{R}$, preserving the integers, and build the ordinary differential equation over \mathbb{R}^3

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

where $r(x)$ is a rounding-like function that has value n whenever $x \in [n - 1/4, n + 1/4]$ for some integer n , $\theta(x)$ is 0 for $x \leq 0$ and $\exp(-1/x)$ for $x > 0$, and c is some suitable constant. A simple analysis of this dynamics shows that it basically alternates $y_1 \leftarrow f(y_2)$ and $y_2 \leftarrow y_1$. As a result, the stroboscopic view of this differential equation is the discrete-time system $y(n+1) = f(y(n))$.

Indeed *dynamic undecidability* constructions have been used in various continuous time models, including piecewise constant maps [31], general dynamical systems [83] and polynomial ordinary differential equations [187].

Observe that dynamic undecidability has merits not covered by static undecidability. Considering Turing machines as dynamical systems provides a view not covered by the von Neumann picture [102]. This also shows that many qualitative features of (analog or non-analog) dynamical systems, e.g. questions about basins of attraction, chaotic behavior or even periodicity, are not computable [257] even for a fixed dynamical system (corresponding for example to a universal Turing machine). Conversely, this brings into the realm of Turing machines and computability general questions traditionally related to dynamical systems [65]. These include, in particular, the relations between universality and chaos [25], necessary conditions for universality [146], computability of entropy [219] and understanding of edge of chaos [227].

One may object that the above simulations require unrealistic hypotheses on systems. For example, that it requires the ability of doing computations with arbitrary precisions: $O(2^{-n})$ precision corresponds roughly to size $O(n)$ of the tape in the simulation above. But at this point, in order not to overinterpret such statements, we

believe that some digression about computational model would be clarifying. This is the topic of the next section.

4 Philosophical, Mathematical and Physics Related Aspects

4.1 Mathematical Models vs Systems

While the use of dynamical systems is very common in experimental sciences (biology, physics, chemistry, etc...), it is important to remember that this is an abstract (mathematical) view, and that the properties of those systems can actually be different in many aspects from those of the initial system they intend to model.

In the literature, there are many accounts in various contexts about situations where models behave very differently according to the levels of modelization. As an illustrative example, we can mention the Lotka Volterra equations discussed in [228, 222]. It is shown in these articles that no classical ordinary differential equation method (unless biased on purpose) behave as stated in classical studies of these equations: no closed cyclic behaviour is observed for example, and “energy” like functions supposed to be preserved are not. While the continuous Lotka Volterra dynamic is usually presented as an abstraction of a discrete setting, these results can be seen as a nonmatching of behaviours between discrete and continuous models.

A context where various semantics have been discussed, with clear stated vocabulary, is that of chemical reactions. To a given formal set of chemical reactions, a hierarchy of semantics can be associated at different levels of abstraction [165].

The most concrete (low-level) interpretation is provided by the *Chemical Master Equation (CME)* which defines the probability of being in a state x at time t by considering the system as a continuous time Markov chain governed by *propensity* of reactions over discrete number of molecules, giving the rate of reactions of the associated chain. The *differential semantics* describes the evolution of the system by an ordinary differential equation (ODE). This ODE can be seen as the mean-field view of the reactions, and can be considered as being derived from the CME by a first-order approximation. Then the *stochastic semantics* is defined by considering a Continuous Time Markov Chain (CTMC) over integer numbers of molecules (discrete concentration levels). The very classical algorithm of Gillespie [182] provides a method to simulate efficiently this CTMC from the reactions. If the simulation provided by this algorithm is often similar to the ODE simulation for large number of molecules, it may exhibit qualitatively different behaviors, in particular when the number of molecules is small [165].

The abstraction of stochastic semantics by simply forgetting the probabilities also yields to the *Petri net semantics* of the reactions, where the discrete states define the number of tokens in each place, and the transitions consume the reactant tokens and produce the product tokens. The abstraction of the Petri net semantics into *Boolean*

semantics is then obtained by reasoning only on the absence/presence of a given molecule.

All these discrete and stochastic trace semantics of reactions systems can be related by a Galois connection in the framework of abstract interpretation [167]. However, it is important to always remember that the behavior of a given semantics can differ from the others in many ways.

The study of the relationship between the differential and the stochastic semantics dates back to the seminal work of Boltzmann in the 19th century who created the domain of Statistical Physics. In this setting, the differential semantics is obtained from the stochastic semantics by limit operations, where the number of molecules tends to infinity and the time steps to zero, under several assumptions such as perfect diffusion [167]. More generally, there is a whole mathematical theory justifying the passage from a stochastic dynamic over a (usually huge) population of agents to its first-order ordinary differential equation description [224]. However, hypotheses of mathematical theorems are not always valid in the context of experimental science applications and concerns are sometimes very different. Notice that the specific context of chemistry has been discussed in articles such as [181].

All these discussions emphasize the difference between a system and its models, and even between the various levels of abstraction of a given system, and the importance of stating explicitly the considered semantics in the coming discussions.

4.2 Church Turing Thesis and Variants

A common statement of the Church Turing Thesis is that *every effective computation can be carried out by a Turing machine, and vice versa*. However, it is often misunderstood, and an instructive discussion demonstrating the improper semantic shift from its initial statement and the way it is often misunderstood today can be found in [133]. We repeat some of these elements here.

Following [133], “The Church Turing thesis concerns the concept of an effective or systematic or mechanical method in logic, mathematics and computer science. ‘Effective’ and its synonyms ‘systematic’ and ‘mechanical’ are terms of art in these disciplines: they do not carry their everyday meaning.”

A myth seems to have arisen progressively in several documents about the fact that Alan Turing’s 1936 paper was establishing facts about limitations of mechanisms [133]. The thesis that *whatever can be calculated by a machine* (working on a finite data in accordance with a finite program of instructions) *is Turing machine-computable* (sometimes referred to as the *Physical Church Turing thesis*, or *Thesis M* of [176]) is a very different statement from the Church Turing thesis.

Thesis M itself admits two interpretations, according to whether the phrase *can be generated by a machine* is taken in the sense *can be generated by a machine that conforms to the physical laws (without resource constraints) of the actual world*, or in a wide sense whether the considered machine could exist in the actual world [133]. Under the latter interpretation, the thesis is clearly wrong, and hence not

really interesting: see all examples of hypercomputations mentioned in next subsection.

The discussion above suggest that it may also be important to distinguish machines from their models. The thesis that *whatever can be calculated by a given model of machine in model(s) of our physical world is Turing machine-computable* might still be one more different statement.

4.3 Are Analog Systems Capable of Hypercomputations?

Several results have shown that analog systems are capable of hypercomputations. One classical way to do so is to consider non-computable reals in the machine (see previous discussion in Section 3.1 and coming Sections 6.1 and 6.2). Another classical way is to use Zeno’s phenomenon: the possibility of simulating an unbounded number of discrete steps in a bounded continuous time (e.g. [30, 258, 62], see section 4.4). A presentation of various systems capable of hypercomputations, with in particular some accounts on analog models of computations, can be found in [321].

Such results may give some clues about the fact that the considered model or systems are indeed very/super powerful. But, as already stated in [242], we believe that one of the main interest of such results is not really a mean to advocate models with hypercomputational power, but rather a way to demonstrate that an extended definition of computation and computability theory including alternative (especially analog) models in addition to the Turing machines are needed [243, 242].

We also personally believe that all variants of the Church Turing Thesis above are actually about “*reasonableness*”: what is a reasonable notion of mechanical method in mathematics and logic for the Church Turing thesis, what is a reasonable machine for Thesis M, or what is a reasonable model of the physics of our world for the last variant. In a contrapositive way, establishing a hyper-computability result is a way to outline what should (possibly) be corrected in the model to make it reasonable [64].

Notice that this has been used for example to advocate for corrections of models from Physics by Warren D. Smith in several papers [313, 314], taking the Church Turing as a postulate, or if one prefers in the context of physics, as a physical law of our physical world.

For accounts on physical limitations against hypercomputational possibilities, one can refer to [125].

4.4 Can Analog Machines Compute Faster?

All previous variants also have effective versions, dealing with complexity, sometimes called the *Effective Church Turing Thesis* or *Strong Church Turing Thesis*

(SCTT). We now come to the question of whether analog machines satisfy these versions.

In 1986, one of the first paper devoted to the explicit question of whether analog machines can be more efficient than digital ones was published [329]. The authors claim that the SCTT is provably true for continuous time dynamical systems described by any Lipschitzian ODE $y' = f(y)$. However, their proof assumes that the time variable remains bounded. No clear arguments is established for the general case, and this was considered as an open problem until very recently. Interesting results and discussions on the subject can also be found in [56].

The question of whether analog machines satisfy SCTT turns out to be deeply related to the question on how resources such as time is measured. In short, the difficulty is that the naive idea of using the time variable of the ODE as a measure of “time complexity” is problematic, since time can be arbitrarily contracted in a continuous system due to the “Zeno phenomenon”. For example, consider a continuous system defined by an ODE $y' = f(y)$ where $f : \mathbb{R} \rightarrow \mathbb{R}$ and $y : \mathbb{R} \rightarrow \mathbb{R}$. Now consider the system

$$\begin{cases} z' = f(z)u \\ u' = u \end{cases}$$

where $u, z : \mathbb{R} \rightarrow \mathbb{R}$. It is not difficult to see that this system rescales the time variable and that its solution is given by $u(t) = e^t$ and $z(t) = y(e^t)$. Therefore, the second ODE simulates the first, with an exponential acceleration.

In a similar manner, it is also possible to present an ODE which has a solution with a component $u : [0, \frac{\pi}{2}) \rightarrow \mathbb{R}$ such that $u(t) = y(\tan(t))$, i.e. it is possible to contract the whole real line into a bounded set. Thus any language computable by the first system (or, in general, by a continuous system) can be computed by another continuous system in time $O(1)$. This problem has been observed or used in many continuous models [296, 297, 258, 62, 63, 12, 101, 143, 131, 132].

Notice that in addition to time contraction, space contraction is also possible, by considering changes on space variables. Note however, that space contractions are more model-dependent and are not always possible. For example, one could consider the system $z(t) = y(t)e^{-t}$, which makes the system exponentially smaller. However, doing so with a polynomial differential equation (for example) requires to add a variable $u(t) = e^t$ to the system, thus making the contraction less useful, if useful at all.

Such time or space contraction phenomena seem, in many systems, to physically correspond to some infinite energy. However, *energy* is not a mathematically universally defined concept, and defining a robust notion of “time complexity” in those systems was an open problem until lately.

It has only recently been shown that analog systems satisfy SCTT, if “time” (as in “time complexity”) is measured by the length of the trajectory [74, 73], and if considering polynomial ordinary differential equations. Since this class of ODEs is very wide and covers in practice most reasonable classes, this can be considered as a definitive proof of the statement that many analog systems satisfy SCTT.

Note that this does not cover all variants of the SCTT, since ODEs do not cover all known physics. In particular, models that rely on quantum effects, General Rela-

tivity, and more generally models that rely on physical experiments (see Section 6.2 for example) do not fall into this class. In other words, this does not cover the “physical” variants of the SCTT.

4.5 Some Philosophical Aspects

Some of the previously mentioned results led to discussions in the context of philosophy.

First, the consideration of systems that could realize hypercomputation gives rise to the following philosophical question: since the physical construction of a hypercomputational model is out of the framework of computability theory, answering the *hypercomputation thesis* requires the construction of a physical model of hypercomputation. It has been argued that, even if such a model is built, it would be impossible to verify that this model is indeed computing a function non-computable by a Turing machine [171, 144]. However, it was pointed out that this objection is not specific to hypercomputation since it is already not possible to verify that a Turing computable function is correct in finite time in general [134]. Furthermore simply because there is no systematic way of computing the values of the halting function does not mean that one could not (in principle) assign one mathematician to each value and ask each of them to come up with a different method, therefore bypassing the problem of uniformity of computation.

The possibility of new computational models which may be exponentially more efficient than any previously known machine (in a wide context, including for example models based on quantum mechanics) suggest alternative, empirical views of complexity theory, usually considered as part of logic and computer science. The arguments in favor of this view and its consequences have been discussed in [275, 276]. It raises some fundamental questions such as whether the empirical limits of computing are identical to limits of algorithms or, to put it differently, what is the capability of symbolic processes to simulate empirical processes. This helps to study the epistemology of computations realized by a machine. In particular, the existence of a function computable by a machine but not by an algorithm would imply that some mathematical problem are solvable by empirical processes without any mathematical method [274, 276].

The statement that there is no equivalent of the Church-Turing thesis for computation over the reals (often relying on the non convergence of formalisms as for discrete computations) is shown in [277] to confound the issue of the extension of the notion of effective computation to the reals on the one hand, and the simulation of analog computers by Turing machines on the other hand. It is possible in both cases to argue in favor of a Church-Turing thesis over the reals [277]. It has also been argued that analog computation literature is often mixing both the concept of continuous valued computations and analog machines, and that a concept called *model-based computation* can help to untangle the misconception, by offering a two

dimensional view of computation: one dimension concerns models and the other the type of variables that are used [43].

Reflexions on whether nature can be considered as computing, and even about the notion of computation have also been discussed [234] in light of new technologies in physics and new ideas in computer sciences such as quantum computing, networks, non-deterministic algorithms.

Some original views on the Church Turing thesis have also been expressed by Dowek [152, 151]. The statement that all physically realized relation can be expressed by a proposition of the language of mathematics (that can be called *Galileo thesis*) can be considered as a consequence of the physical Church thesis [152].

5 Theory of Analog Systems

We now review the various computational theories that have been developed for analog systems.

5.1 Generic Formalizations of Analog Computations

There have been several approaches to formalize analog computations in a general setting.

5.1.1 Formalization by Abstract State Machine

A formalization of generic algorithms covering both analog and classical, discrete and continuous time algorithms has been proposed in [70]. This framework, extending [69], is based on an extension and reformulation of *abstract state machines* (ASM) from Gurevich [196]. The notion of analog ASM program is introduced, and a completeness result is proven: any process satisfying the three postulates generalizing those of [196] is demonstrated to correspond to some analog ASM program [70]. This is intended to be a first step towards a formalization and a proof of an equivalent of a Church Turing thesis for analog continuous and discrete time systems in the spirit of what has been achieved for discrete-time models [59, 147, 60].

5.1.2 Formalization by Fixed Point Techniques

An approach to define computations by analog models in general is to consider networks of analog modules connected by channels [326], processing data from a metric space A , and operating with respect to a global continuous clock \mathbb{T} . The inputs and outputs of the network are continuous streams ($C = \mathbb{T} \rightarrow A$) and the

input-output behaviour of the network, usually with external parameters, is modeled by a function $\phi : C^r \times A^p \rightarrow C^q$. The authors focus on the important case where the modules are *causal*, that is the output at time t only depends on the inputs over $[0, t]$.

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

A general framework to deal with fixed point techniques in analog systems, based on Fréchet spaces, has recently been developed in [284].

An abstract model of computability over data stream using a high-level programming language, an extension of “while” programs, over abstract data-types (multi-sorted algebras) has been considered in [327]. The authors analyze when concrete and abstract models are equivalent.

5.1.3 Formalization and Proof Theory for Cyberphysical Systems

A whole theory for the analysis, logic and proofs for cyberphysical systems has been developed [283]. Rich logical theories to cover both continuous differential equation dynamics and discrete changes involved in cyberphysical or hybrid systems have been proposed and proven to be sound and complete [278, 280, 281, 280]. These logical foundations, built incrementally in a series of articles, are now forming an elegant proof-theoretical bridge aligning the theory of continuous systems with the theory of discrete systems [279, 282]. They are the basis of theorem provers *KeYmaera* and *KeYmaera X*, which demonstrated their impact on various concrete applications. For an overview of all the existing works along these lines of research at both theoretical and practical levels, refer to [282, 283].

5.2 \mathbb{R} -recursion Theory

\mathbb{R} -recursive functions were introduced as a theory of recursive functions on the reals built in analogy with classical recursion theory to deal with conceptual analog computers operating in continuous time [258]. Moore considered the smallest class of functions obtained from constants 0 and 1, projections and closed by composition, integration and minimization. He demonstrated that various non-recursively enumerable sets fall into the proposed hierarchy, defined according to the number of minimizations involved.

The initial theory [258] suffers from some mathematical difficulties when considering several nested minimization operators, but its foundational ideas gave birth to several lines of research, both at the computability and complexity level.

At the computability level, since minimization is the operation that gives rise to uncomputable functions, and difficulties, a natural question is to ask if it can be replaced by some other natural and better defined operator of mathematical analysis. This can be done by replacing minimization by some limit operation [262]. In addition, it is shown that the obtained hierarchy does not collapse [232, 231], which implies that infinite limits and first order integration are not interchangeable operations [136]. A presentation and overview of the obtained real recursive functions theory is presented in [137].

The algebra of functions built without the minimization operator only contains analytic functions and is not closed under iteration [105]. Closure by bounded products, bounded sums and bounded recursion has also been investigated.

However, if an arbitrarily smooth extension θ to the reals of the Heaviside function is included in the set of basic functions, then extensions to the reals of all primitive recursive functions are obtained. More generally, several authors studied extensions of the algebra of functions by considering an integration operator restricted to *linear* differential equations, and considering that such an arbitrarily smooth extension θ was among the basic functions. The obtained class contains extensions to the reals of all the elementary functions [102, 106, 103, 104]. By adding a suitable basic exponential iterate, extensions to the reals of all the functions of the classes of the Grzegorzczuk's hierarchy can also be obtained.

Instead of asking which computable functions over \mathbb{N} have extensions to \mathbb{R} in a given function algebra, one can consider classes of functions over \mathbb{R} computable according to recursive analysis, and characterize them precisely with function algebras. This was done for elementarily computable functions [75], considering a restricted limit schema, and for computable functions [76], considering a restricted minimization schema.

A generic framework for presenting previous results, based on the notion of approximation, has been developed in [107].

At the complexity level, characterizations of complexity classes such as P and NP using \mathbb{R} -recursive algebra have been obtained [264]. This is done by ensuring that at every step of the construction of a function in previous classes, each component cannot grow faster than a quasi-polynomial. This allows to transfer classical questions from complexity theory to the context of real and complex analysis [233, 264, 263].

5.3 Analog Automata Theories

Several lines of research have been devoted to adapt classical discrete automata theory to analog domains or continuous time.

Rabinovich and Trakhtenbrot have developed a *continuous time automata* theory in [289, 324, 288]. In this approach, automata are not considered as language recognizers, but as computing operators on signals, that is on functions from the non-negative real numbers to a finite alphabet, seen as channel's states. For a presentation as well as extensions, see [172].

Another approach has been considered, where ODEs equipped with a tape-like function memory are used to recognize sets of piecewise continuous functions, yielding a Chomsky-like hierarchy [299].

A class of recognizable functions extending stochastic and quantum functions, and with a cutpoint theorem similar to the one for probabilistic automata theory and with nice closure properties has been determined in [82].

Topological automata were introduced as a generalization of previous definitions of probabilistic and quantum automata [214]: deterministic or nondeterministic probabilistic and quantum automata are proven to recognize only regular languages with an isolated threshold. A topological theory of continuous-time automata has also been developed [250]: the basic idea is to replace finiteness assumptions in the classical theory of finite automata by compactness assumptions. Some existence results and a Myhill-Nerode theorem is obtained in this framework covering both finite automata and continuous automata.

Generalized finite automata working over arbitrary structures have been introduced [175]. Structural properties of accepted sets as well as computational hardness of classical questions for this model have been investigated [247]. A restricted version of the model has also been considered [248]: it is demonstrated to satisfy a pumping lemma, and to yield decidability results closer to what would be expected for a notion of finite automata over arbitrary structures, such as the reals.

All previous approaches are not easily connected, and are rather independent views and models.

6 Analysing the Power and Limitations of Analog Computations

6.1 Neural Networks

The computational power of artificial neural networks has been investigated intensively in the 90's in many papers: see the instructive survey [311] and more recently [92]. Finite analog recurrent networks with integer weights are known to be essentially equivalent to finite automata, while finite analog recurrent networks with rational weights can simulate arbitrary Turing machines step by step [309]. When considering arbitrary real weights (possibly non-computable) one obtains the non-uniform versions of the associated complexity and computability classes. In particular, one gets models that can decide predicates or sets that are not computable by Turing machines [308]. For a full discussion, see [307]. Improvements on the mapping from a Turing machine to a recurrent artificial neural network have recently been proposed [111]. It has also been proven that the question $P = ?NP$ relativizes naturally to similar questions related to artificial neural networks with real weights [135].

Previous results have been extended to the case of evolving recurrent neural networks, i.e., networks where the synaptic weights can be updated at each discrete

time step [93, 95]. Interactive recurrent neural networks have been considered in [94, 97].

When subjected to some infinite binary input stream, the Boolean output cells necessarily exhibit some attractor dynamics. A characterization of the expressive powers of several models of Boolean, sigmoidal deterministic, and sigmoidal non-deterministic first-order recurrent neural networks, in relation with their attractor dynamics has been established in [99, 98, 100]. This extends results about their expressive power characterized in terms of topological classes in the Borel hierarchy over the Cantor space [96].

The computational power of spiking neuron models has been investigated in a series of papers: refer to [311, 236] or [179] for surveys.

6.2 Physical Oracles

In a recent series of papers, started by [45], Beggs, Costa, Poças and Tucker consider various dynamical systems doing computations involving discrete data and physical experiments. Such systems can be modeled by Turing machines with oracles that correspond to physical processes. A good summary of their results can be found in [15]. They analyze the computational power of these machines and more specifically the limits of polynomial time computations. They show that for a broad class of physical oracles, an upper bound on polynomial time computations is the non-uniform complexity class² $BPP//\log\star$, and that $P/poly$ can be attained by using non-computable analog-digital interface protocols.

Without giving too much details, in this model the Turing machine has access to a physical oracle, such as [46]. Whenever the machine wants to call the oracle, it writes some input on the oracle tape and enters a special oracle state. The machine is then suspended and a physical experiment starts. Importantly, to each machine is associated a time schedule function T , usually a time-constructible function. If the experiment finishes in time less than $T(|z|)$, where z is the content of the oracle tape, the machine resumes in one of finitely many states encoding the outcome of the experiment (typically *YES* or *NO*). Otherwise, the experiment is stopped and the machine resumes in a special state *TIMEOUT*.

The authors analyzed many kinds of physical oracles and identified three major forms of experiments. In most cases, the experiment encodes a single real number

² Let \mathcal{B} is a class of sets and \mathcal{F} a class of functions. The advice class \mathcal{B}/\mathcal{F} is the class of sets A such that there exists $B \in \mathcal{B}$ and $f \in \mathcal{F}$ such that for every word w , $w \in A$ if and only if $\langle w, f(|w|) \rangle \in B$. We use \log to denote the class of functions f such that $|f(n)| = \mathcal{O}(\log n)$ as $n \rightarrow \infty$. We use $poly$ to denote the class of functions f such that $|f(n)| = \mathcal{O}(n^k)$ for some k as $n \rightarrow \infty$. Thus $P/poly$ corresponds to a non-uniform polynomial size advice. The class $BPP//\mathcal{F}\star$ is the class of sets A such that there is a probabilistic polynomial time Turing machine \mathcal{M} , a prefix function $f \in \mathcal{F}\star$ (i.e. $f(n)$ is a prefix of $f(n+1)$) and a constant $\gamma < \frac{1}{2}$ such that for every word w , on input $\langle w, f(|w|) \rangle$, \mathcal{M} rejects with probability at most γ if $w \in A$, and accepts with probability at most γ if $w \notin A$. See [45] for more details on those classes.

a and the oracle tape encodes a rational x . Each class of experiments specifies not only the behavior of the experiments but also how long it takes to complete.

- **Two-Sided Case:** the experiment takes time $t(x, a) = C/|x - a|^d$ for $x \neq a$ and answers YES if $x < a$ and NO if $x > a$. It always times out if $a = x$. A typical example is a scale, where the time needed to detect movement to either side is inversely exponential to the difference between the masses.
- **Threshold Case:** the experiment takes time $t(x, a) = C/|x - a|^d$ for $x > a$ and always answers YES. It always times out if $x \leq a$. Several examples including the photoelectric effect and Rutherford scattering are given by the authors.
- **Vanishing Case:** the experiment can only detect if $x \neq a$ and times out if $x = a$. However, given two experiments x, x' , we can detect which of x and x' is closer to a . Informally, the reader can think of an experiment where we can measure $\alpha(x - a)^2$ (with α some unknown parameter): we cannot detect the sign of $x - a$ but for two experiments x, x' , α are the same so we can compare $\alpha(x - a)^2$ to $\alpha(x' - a)^2$. Examples include some modified Wheatstone bridge and Brewster's angle experiment.

Orthogonally to the type of physical experiment, the authors also study variants on how precise the experiment is with respect to x . In the *infinite* precision case, the experiment is done exactly with x . In the *unbounded* precision case, the experiment is done with $x' \in [x - \varepsilon, x + \varepsilon]$ (drawn uniformly at random and experiments are independent) where ε is part of the input and can be arbitrarily small (but not zero). In the *fixed* precision case, ε is a constant of the experiment and cannot be changed.

More recently, the authors have developed a hierarchy for BPP//log \star based on counting calls to a non-deterministic physical oracle modeled by a random walk on a line [44].

A general framework for describing physical computations, covering above approaches has been proposed in [334].

6.3 On the Effect of Noise on Computations

The invariant (ergodic) measure of dynamical systems has been proven to remain computable even if a small amount of noise is introduced in the system [84]. This demonstrates that while some dynamical systems have been proven to simulate Turing machines, the presence of noise forbids uncomputability. Actually, when considering a compact domain, it has been proven that analog neural nets with gaussian or other common noise distributions cannot even recognize arbitrary regular languages [237, 238].

Some tight bounds on the space-complexity of computing the ergodic measure of a low-dimensional discrete-time dynamical systems affected by a Gaussian noise have been obtained [85].

Previous results have implied some limitations on the ability of physical systems to perform computations, namely on their ability to store information. Memory is

roughly defined as the maximal amount of information that the evolving system can carry from one instant to the next, and is demonstrated not to be able to grow too fast.

This is discussed to be in favor of the following postulate for physical computations. If a physical system has memory s available to it, then it is only capable of performing computations in the complexity class $SPACE(poly(s))$ even when provided with unlimited time [86]. This has been proven to hold for several classes of dynamics and noise models over a compact domain.

6.4 Complexity theories for Analog Computations

There have been several attempts to build a complexity theory for continuous time systems (but not intended to cover generic ODEs).

This includes the theory where the global minimizers of particular energy functions are supposed to give solutions of some problem [184, 185]. The structure of such energy functions leads to the introduction of problem classes U and NU , with the existence of complete problems for these classes.

Another attempt is focused on a very specific type of systems: *dissipative flow models* [50]. This theory has been used in several papers from the same authors to study a particular class of flow dynamics [49] for solving linear programming problems.

Neither of the previous two approaches is intended to cover generic ODEs, and none of them is able to relate the obtained classes to classical classes of computational complexity, unlike the approach presented in Section 7.

6.5 Chemical Reaction Networks

Recent years have seen an important literature about the computational power of *chemical reaction networks* (CRN) [316, 130]. The model is built by analogy with concrete chemistry, see also [291] for a presentation. A program corresponds to a set of chemical rules over a finite set of species, abstracting from the matter conservation laws, considering well-mixed solutions, and ignoring spatial properties of the molecules. They can be seen as natural extensions of population protocols discussed in Section 2.8: here reactions are not assumed to be between pairs of agents/molecules, and reaction rates can possibly be considered.

One can distinguish various dichotomies of computations in CRNs [141, 315]: the number of molecules can either be considered as *discrete* or *continuous*. Computations can be considered either *uniform* (the same CRN handles all inputs) or *non-uniform* (for each input size, a different CRN is considered). In the *deterministic* setting the output is considered to be guaranteed, while in the *probabilistic* setting some probability of error is considered. The *halting approach* considers notions of

acceptance where agents irreversibly produce an answer, while in the *stabilizing approach* the population eventually stabilizes to an answer, in the spirit of the acceptance criteria for population protocols.

CRNs have clear connexions to various other models, including Petri nets, and vector addition systems [130]. These models can be classified as *unordered* with respect to classical models of computability such as Turing machines where an order on instructions is considered. The presence of reaction rates or probabilities is a way to provide back some order on instructions, while forgetting order in classical models of computability gives rise to similar models [130].

This abstract model is also motivated by concrete implementations using an experimental technique called DNA strand displacement [317, 108, 287, 121]. Implementations with proteins are also considered [166].

The computational power of CRNs over many of these variants has been studied in several papers [316, 119, 141]. It has been proven, using in particular the strong technical result from [162], that chemical reaction networks turn out to be a robust model from a computability point of view [88] in the sense that variations on the notion of output conventions for error-free computations basically yields either Turing machines or a model whose power is equivalent to population protocols discussed in Section 2.8, that is to say whose computational power is limited to semi-linear sets.

Lower bounds on the time required to build rules that would act as a “timer” have been obtained [150], answering the open question on the ability of population protocols to perform a fast leader election in the probabilistic settings, as well as their ability to simulate arbitrary algorithms from a uniform initial state.

The case of the continuous settings with the usual natural semantics has been an open problem before being very recently solved [166] using the theory discussed in Section 7. The computational power of a variant where reaction rates are abstracted was previously characterized [120].

7 Computations by Polynomial Ordinary Differential Equations

Polynomial ordinary differential equations have been shown to be closely related to the GPAC of Shannon discussed in Section 2.2.1 and to have a very rich theory. In particular, they may now be considered for continuous time models as the equivalent of Turing machines for discrete time models in many aspects.

7.1 GPAC and Polynomial Ordinary Differential Equations

The GPAC, presented in Section 2.2.1, can also be presented in terms of polynomial ordinary differential equations. More precisely, y is *generable by a GPAC* if and only if there exist functions $z = (z_1, \dots, z_n)$, a vector of polynomials p and an initial

$$z' = az^2 + bz + c, \quad P = az^2 + bz + c - z'$$

$$z'' = 2az + b$$

$$z''' = 2a$$

A Survey on Analog Models of Computation.

condition z_0 such that

$$y \equiv z_1, \quad z(0) = z_0, \quad \text{and} \quad \underline{z' = p(z)}.$$

In other words, y is a component of the solution of a system of polynomial differential equations. One advantage of this presentation is that it eliminates all the problems related to the domain of definition and the solution is necessarily unique. In particular, the solutions are always analytic functions.

This fact actually has a complicated history because of gaps in proof found by various authors and the refinements in the definition of the GPAC: Shannon, while introducing his model [305], proved that any function y generated by a GPAC is solution of a polynomial Differential Algebraic Equation (pDAE), that is there exists a polynomial p such that

$$p(y, y', \dots, y^{(n)}) = 0 \quad (3)$$

for some integer n . The converse inclusion was claimed by Shannon, which led some authors to define the GPAC in terms of differential equations directly [286]. However this equivalence requires some care because of the domain of definition, and furthermore, DAEs do not necessarily have a unique solution, which complicates this approach. It was later realized that unary functions generated by a GPAC (with some restrictions to fix earlier problems) are exactly components of polynomial Initial Value Problems (pIVP) [188]. Several variations on the GPAC circuits were explored and proven to be all equivalent, which essentially shows that the above notion is robust and probably the right definition to be considered [186].

A famous statement, due to Rubel [294], shows that polynomial Differentially Algebraic Equations (pDAEs) are universal for continuous functions. More precisely, there is a universal pDAE of order 4 whose solutions can approximate any continuous function with arbitrary precision [294]. This work has recently been improved [140] to hold for order 3. It has been established very recently that this also holds for polynomial ordinary differential equations [77].

7.2 GPAC Generable Functions

The class of generable functions is particularly robust because if f and g are generable then $f \pm g$, fg , $\frac{f}{g}$ and $f \circ g$ are also generable. Moreover, if f is generable and y satisfies the differential equation $y' = f(y)$ then y is also generable. Shannon's original work was intended for circuits with several inputs, but this was never formalized. Recent work provides a proper description of the class of generable functions over multi-dimensional domains and shows that it also enjoys many closure properties [72].

z' is a polynomial function of z .
Does this reduce to D-finite?

guess: YES
 $\binom{n}{k}$

$$\sum_{k=0}^n f(k) \cdot g(n-k)$$

$$\begin{array}{c} k \quad n-k \\ | \quad | \end{array}$$

7.3 GPAC computability

The fact that the functions generated by the GPAC (i.e. generable functions) are analytic (or C^∞ for the more relaxed models, without external inputs) has historically been seen as a limitation of the GPAC.

This has been proven to be an artefact of the model and an alternative notion of computability based on polynomials ODEs has been proposed [66]. In fact the authors prove that the GPAC and Turing machines are equivalent. The fundamental idea is that while generable functions correspond to “*real-time computability*” (i.e., the system computes the answer instantaneously), considering a notion similar to the one used for Turing machines (i.e., the system evolves and “converges effectively” to the answer) yields a computational power equivalent to that of Turing machines.

This equivalence between the GPAC and Computable Analysis can be reformulated as follows:

Theorem 7.1 (Equivalence between GPAC and Computable Analysis, [66]). *A function $f : [a, b] \rightarrow \mathbb{R}$ is computable (in the sense of Computable Analysis) if and only if there exists an integer d and a vector of polynomials $p : \mathbb{R}^d \rightarrow \mathbb{R}^d$ with rational coefficients such that for any $x \in [a, b]$, the unique solution $y : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}^d$ to*

$$y(0) = (0, \dots, 0, x), \quad y'(t) = p(y(t))$$

satisfies that

$$|f(x) - y_1(t)| \leq y_2(t) \quad \text{and} \quad y_2(t) \rightarrow 0 \text{ as } t \rightarrow \infty.$$

In other words, a function f is computable if there is a GPAC that, on input x , has one of its components (y_1) converge to $f(x)$. Not only it converges, but another component (y_2) gives a bound on the error between $f(x)$ and y_1 .

This unexpected connection between Turing computability and the GPAC was recently refined [74, 73] at the level of complexity, with a characterization of the class P and polynomial time computable real functions.

A difficult point in the context of analog computability, and in the GPAC in particular is to define a notion of complexity that makes sense and is sufficiently robust: see discussions in Section 4.4. In fact, the intuitive notion of measuring the complexity based on the convergence rate (i.e. how fast $y_1(t) \rightarrow f(x)$) does not work.

One recently proposed solution to this problem is to measure the complexity using the length of the curve y instead of time. This process is illustrated in Figure 4. We recall that the length of a curve $y \in C^1(I, \mathbb{R}^n)$ defined over some interval $I = [a, b]$ is given by $\text{len}_y(a, b) = \int_I \|y'(t)\|_2 dt$,

Theorem 7.2 (Equivalence between GPAC and CA (Complexity), [73]). *A function $f : [a, b] \rightarrow \mathbb{R}$ is computable in polynomial time (in the sense of Computable Analysis) if and only if there exists a polynomial $L : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$, an integer d and a vector of polynomials $p : \mathbb{R}^d \rightarrow \mathbb{R}^d$ with coefficients in \mathbb{Q} , such that for any $x \in [a, b]$, the unique solution $y : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}^d$ to*

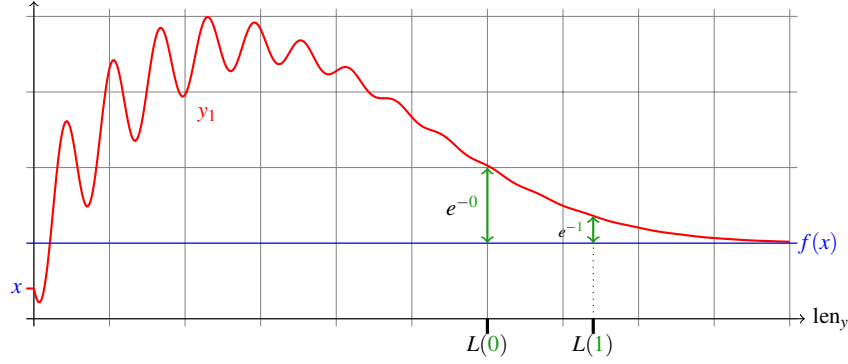


Fig. 4: Graphical representation of analog computability (Theorem 7.2): on input x , starting from initial condition $(x, 0, \dots, 0)$, the polynomial ordinary differential equation $y' = p(y)$ ensures that $y_1(t)$ gives $f(x)$ with accuracy better than $e^{-\mu}$ as soon as the length of y (from 0 to t) is greater than $L(\mu)$. Note that we did not plot the other variables y_2, \dots, y_d and the horizontal axis measures the length of y (instead of the time t).

$$\underline{y(0) = (x, 0, \dots, 0)}, \quad \underline{y'(t) = p(y(t))}$$

satisfies for all $t \in \mathbb{R}_{\geq 0}$:

- for any $\mu \in \mathbb{R}_{\geq 0}$, if $\text{len}_y(0, t) \geq L(\mu)$ then $|f(x) - y_1(t)| \leq e^{-\mu}$,
- $\|y'(t)\| \geq 1$.

In other words, the precision of y_1 increases with the length of the curve. More precisely, as soon as the length between 0 and t is at least $L(\mu)$, the precision is at least $e^{-\mu}$. Notice how rescaling the curve would not help here since it does not change the length of y . The second condition on the derivative of y prevents some pathological cases and ensures that curve has infinite length, and thus that y_1 indeed converges to $f(x)$. It is possible to extend this equivalence to multivariate functions and unbounded input domains such as \mathbb{R} , by making L take into account the norm of x .

It also possible to define the class P directly in terms of differential equations, by encoding words with rational numbers. Again the length plays a crucial role, but since a differential equation does not “stop”, the component y_1 is used to signal that it accepts ($y_1 \geq 1$) or rejects ($y_1 \leq -1$). Figure 5 illustrates this process.

Theorem 7.3 (Analog characterization of P, [73]). *A language $\mathcal{L} \subseteq \{0, 1\}^*$ belongs to P , the class of polynomial time decidable languages, if and only if there exist a polynomial $L : \mathbb{N} \rightarrow \mathbb{N}$, an integer d and a vector of polynomials $p : \mathbb{R}^d \rightarrow \mathbb{R}^d$ with coefficients in \mathbb{Q} , such that for all words $w \in \{0, 1\}^*$, the unique solution $y : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}^d$ to*

$$y(0) = (0, \dots, 0, |w|, \psi(w)), \quad y'(t) = p(y(t))$$

where $\psi(w) = \sum_{i=1}^{|w|} w_i 2^{-i}$, satisfies for all $t \in \mathbb{R}_{\geq 0}$:

- if $|y_1(t)| \geq 1$ then $|y_1(u)| \geq 1$ for all $u \geq t \geq 0$ (and similarly for $|y_1(t)| \leq -1$),
- if $w \in \mathcal{L}$ (resp. $\notin \mathcal{L}$) and $\text{len}_y(0, t) \geq L(|w|)$ then $y_1(t) \geq 1$ (resp. ≤ -1),
- $\|y'(t)\| \geq 1$.

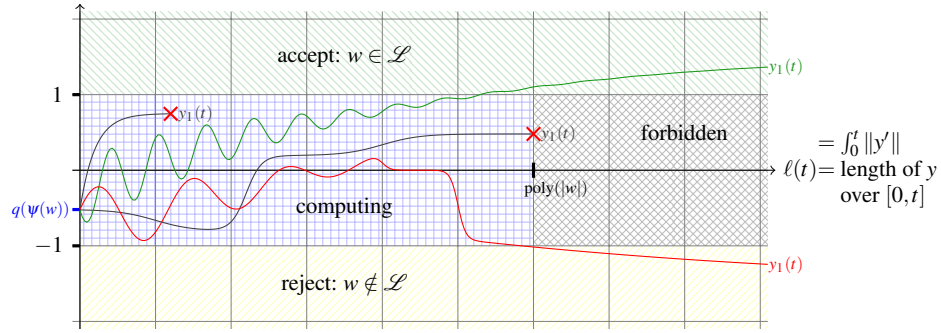


Fig. 5: Graphical representation of the analog characterization of P (Theorem 7.3). The green trajectory represents an accepting computation, the red a rejecting one, and the gray are invalid computations. An invalid computation is a trajectory that is too slow (thus violating the technical condition), or that does not accept/reject in polynomial length. Note that we only represent the first component of the solution, the other components can have arbitrary behaviors.

One clear interest of the previous statements is that they provide a way to define classical concepts from the theory of computation (computable function, polynomial time computable functions) only using concepts from analysis, namely polynomial ordinary differential equations.

8 Acknowledgements

We would like to thank several people for various feedbacks, and many helpful comments: Andrew Adamatzky, Amir Ali Amhadi, Cameron Beebe, Jérémie Cabessa, Liesbeth Demol, Dave Doty, Laurent Doyen, Jérôme Durand-Lose, Marie-Jo Durand-Richard, Javier Esparza, François Fages, Goran Frehse, Laurent Fribourg, Daniel Graça, Eric Goubault, Emmanuel Hainry, Emmanuel Jeandel, Manuel Lameiras Campagnolo, Jérôme Leroux, Jack Lutz, Wolfgang Maass, Klaus Meer, Pekka Orponen, André Platzer, Maël Pégny, Cristóbal Rojas, Keijo Ruohonen, Zoltan Toroczkai, John Tucker, Bernd Ulmann, Erik Winfree, Jeffery Zucker, as well as two anonymous referees.

Special thanks to Ulf Hashagen for many references related to history of computing, Andrew Adamatzky and Keijo Ruohonen for several references and comments, sometimes repeated literally in this document, about some physical or unconventional computing models.

Many thanks to Jérémie Cabessa, Jérôme Durand-Lose and Klaus Meer for many feedbacks including typographic issues about a preliminary version of this document.

References

1. URL <http://www.analogmuseum.org/>
2. URL <https://www.tensorflow.org/>
3. Aaronson, S., Watrous, J.: Closed timelike curves make quantum and classical computing equivalent. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* **465**(2102), 631–647 (2009)
4. Adamatzky, A.: Physarum machine: implementation of a Kolmogorov-Uspensky machine on a biological substrate. *Parallel Processing Letters* **17**(04), 455–467 (2007)
5. Adamatzky, A.: *Atlas of Physarum Computing*. World Scientific (2015)
6. Adamatzky, A., Costello, B.D.L., Asai, T.: *Reaction-diffusion computers*. Elsevier (2005)
7. Adleman, L.M.: Molecular computation of solutions to combinatorial problems. *Science* **266**, 1021–1024 (1994) *DNA computing*
8. Ahmadi, A.A., Jungers, R.M.: Switched stability of nonlinear systems via sos-convex Lyapunov functions and semidefinite programming. In: *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, pp. 727–732. IEEE (2013)
9. Ahmadi, A.A., Jungers, R.M.: On complexity of Lyapunov functions for switched linear systems. *IFAC Proceedings Volumes* **47**(3), 5992–5997 (2014)
10. Ahmadi, A.A., Majumdar, A., Tedrake, R.: Complexity of ten decision problems in continuous time dynamical systems. In: *American Control Conference (ACC), 2013*, pp. 6376–6381. IEEE (2013)
11. Ahmadi, A.A., Parrilo, P.A.: Stability of polynomial differential equations: Complexity and converse Lyapunov questions. *arXiv preprint arXiv:1308.6833* (2013)
12. Alur, R., Dill, D.L.: Automata for modeling real-time systems. In: M. Paterson (ed.) *Automata, Languages and Programming, 17th International Colloquium, ICALP90*, Warwick University, England, July 16–20, 1990, *Proceedings, Lecture Notes in Computer Science*, vol. 443, pp. 322–335. Springer (1990)
13. Alur, R., Dill, D.L.: A theory of timed automata. *Theoretical Computer Science* **126**(2), 183–235 (1994). *Fundamental Study*
14. Alur, R., Henzinger, T.A., Lafferriere, G., Pappas, G.J.: Discrete abstractions of hybrid systems. *Proceedings of the IEEE* **88**(7), 971–984 (2000)
15. Ambaram, T., Beggs, E., Costa, J.F., Poças, D., Tucker, J.V.: An analogue-digital model of computation: Turing machines with physical oracles. In: *Advances in Unconventional Computing*, pp. 73–115. Springer (2017)
16. Ames, A.D., Sastry, S.: Characterization of Zeno behavior in hybrid systems using homological methods. In: *Proceedings of the 2005, American Control Conference, 2005.*, pp. 1160–1165. IEEE (2005)
17. Ames, A.D., Zheng, H., Gregg, R.D., Sastry, S.: Is there life after Zeno? taking executions past the breaking (Zeno) point. In: *2006 American control conference*, pp. 6–pp. IEEE (2006)
18. Andr  ka, H., Madar  sz, J.X., N  meti, I.: *Logical Axiomatizations of Space-Time. Samples from the Literature*, pp. 155–185. Springer US, Boston, MA (2006)
19. Andrew, A.: *Physarum Machines: Computers From Slime Mould*, vol. 74. World Scientific (2010)

20. Angluin, D., Aspnes, J., Chan, M., Fischer, M.J., Jiang, H., Peralta, R.: Stably computable properties of network graphs. In: V.K. Prasanna, S. Iyengar, P. Spirakis, M. Welsh (eds.) *Distributed Computing in Sensor Systems: First IEEE International Conference, DCOSS 2005*, Marina del Rey, CA, USE, June/July, 2005, Proceedings, *Lecture Notes in Computer Science*, vol. 3560, pp. 63–74. Springer-Verlag (2005)
21. Angluin, D., Aspnes, J., Diamadi, Z., Fischer, M.J., Peralta, R.: Computation in networks of passively mobile finite-state sensors. In: *Twenty-Third ACM Symposium on Principles of Distributed Computing*, pp. 290–299. ACM Press (2004)
22. Angluin, D., Aspnes, J., Eisenstat, D., Ruppert, E.: The computational power of population protocols. *Distributed Computing* **20**(4), 279–304 (2007)
23. Angluin, D., Aspnes, J., Fischer, M.J., Jiang, H.: Self-stabilizing population protocols. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)* **3**(4), 13 (2008)
24. Anstreicher, K.M.: Linear programming and the Newton barrier flow. *Mathematical Programming* **41**(1-3), 367–373 (1988)
25. Asarin, E.: Chaos and undecidability (draft) (1995). Available in <http://www.liafa.jussieu.fr/~asarin/>
26. Asarin, E.: Equations on timed languages. In: T.A. Henzinger, S. Sastry (eds.) *Hybrid Systems: Computation and Control, First International Workshop, HSCC'98*, Berkeley, California, USA, April 13-15, 1998, Proceedings, *Lecture Notes in Computer Science*, vol. 1386, pp. 1–12. Springer (1998)
27. Asarin, E., Caspi, P., Maler, O.: A Kleene theorem for timed automata. In: *Proceedings, 12th Annual IEEE Symposium on Logic in Computer Science*, pp. 160–171. IEEE Computer Society Press, Warsaw, Poland (1997)
28. Asarin, E., Caspi, P., Maler, O.: Timed regular expressions. *Journal of the ACM* **49**(2), 172–206 (2002)
29. Asarin, E., Dima, C.: Balanced timed regular expressions. *Electronic Notes in Theoretical Computer Science* **68**(5) (2002)
30. Asarin, E., Maler, O.: Achilles and the tortoise climbing up the arithmetical hierarchy. *Journal of Computer and System Sciences* **57**(3), 389–398 (1998)
31. Asarin, E., Maler, O., Pnueli, A.: Reachability analysis of dynamical systems having piecewise-constant derivatives. *Theoretical Computer Science* **138**(1), 35–65 (1995)
32. Asarin, E., Mysore, V.P., Pnueli, A., Schneider, G.: Low dimensional hybrid systems—decidable, undecidable, don't know. *Information and Computation* **211**, 138–159 (2012)
33. Aspnes, J., Ruppert, E.: An introduction to population protocols. In: *Bulletin of the EATCS*, vol. 93, pp. 106–125 (2007)
34. Baartse, M., Meer, K.: Real interactive proofs for vpspace. In: *LIPICs-Leibniz International Proceedings in Informatics*, vol. 58. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2016)
35. Baartse, M., Meer, K.: An algebraic proof of the real number PCP theorem. *Journal of Complexity* **40**, 34–77 (2017)
36. Basu, S.: A complex analogue of toda's theorem. *Foundations of Computational Mathematics* **12**(3), 327–362 (2012)
37. Basu, S., Zell, T.: Polynomial hierarchy, betti numbers, and a real analogue of toda's theorem. *Foundations of Computational Mathematics* **10**(4), 429–454 (2010)
38. Bayer, D., Lagarias, J.C.: Karmarkar's linear programming algorithm and Newton's method. *Mathematical Programming* **50**(1-3), 291–330 (1991)
39. Bazille, H., Bournez, O., Goma, W., Pouly, A.: On the complexity of bounded time reachability for piecewise affine systems. *Theoretical Computer Science* (2016)
40. Beal, J., Dulman, S., Usbeck, K., Viroli, M., Correll, N.: Organizing the aggregate: Languages for spatial computing. corr, abs/1202.5509, 2012
41. Beauquier, D.: Pumping lemmas for timed automata. In: M. Nivat (ed.) *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, *Lecture Notes in Computer Science*, vol. 1378, pp. 81–94. Springer (1998)

42. Becchetti, L., Bonifaci, V., Dirnberger, M., Karrenbauer, A., Mehlhorn, K.: Physarum can compute shortest paths: convergence proofs and complexity bounds. In: *International Colloquium on Automata, Languages, and Programming*, pp. 472–483. Springer (2013)
43. Beebe, C.: *Model-based computation*. Natural Computing (2017), URL <https://doi.org/10.1007/s11047-017-9643-0>
44. Beggs, E., Cortez, P., Costa, J.F., Tucker, J.V.: A Hierarchy for BPP//log \star Based on Counting Calls to an Oracle, pp. 39–56. Springer International Publishing, Cham (2017)
45. Beggs, E., Costa, J., Loff, B., Tucker, J.: Computational complexity with experiments as oracles. *Proceedings of the Royal Society of London-A* **464**(2098), 2777 (2008)
46. Beggs, E.J., Tucker, J.V.: Experimental computation of real numbers by Newtonian machines. In: *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 463, pp. 1541–1561. The Royal Society (2007)
47. Bell, P.C., Chen, S., Jackson, L.: On the decidability and complexity of problems for restricted hierarchical hybrid systems. *Theoretical Computer Science* **652**, 47–63 (2016)
48. Ben-Amram, A.M.: Mortality of iterated piecewise affine functions over the integers: Decidability and complexity. In: *STACS*, pp. 514–525 (2013)
49. Ben-Hur, A., Feinberg, J., Fishman, S., Siegelmann, H.T.: Probabilistic analysis of a differential equation for linear programming. *Journal of Complexity* **19**(4), 474–510 (2003), URL [http://dx.doi.org/10.1016/S0885-064X\(03\)00032-3](http://dx.doi.org/10.1016/S0885-064X(03)00032-3)
50. Ben-Hur, A., Siegelmann, H.T., Fishman, S.: A theory of complexity for continuous time systems. *Journal of Complexity* **18**(1), 51–86 (2002)
51. Bennett, C.H.: Logical reversibility of computation. *IBM Journal Research Development* **6**, 525–532 (1973)
52. Bennett, C.H.: The thermodynamics of computation—a review. *International Journal of Theoretical Physics* **21**(12), 905–940 (1982)
53. Benveniste, A., Bourke, T., Caillaud, B., Pouzet, M.: Non-standard semantics of hybrid systems modelers. *Journal of Computer and System Sciences* **78**(3), 877–910 (2012)
54. Berry, G., Boudol, G.: The chemical abstract machine. *Theoretical Computer Science* **96** (1992)
55. Bishop, C.M.: *Neural networks for pattern recognition*. Oxford university press (1995)
56. Blakey, E.W.: A model-independent theory of computational complexity: from patience to precision and beyond. Ph.D. thesis, University of Oxford (2010)
57. Blum, L., Cucker, F., Shub, M., Smale, S.: *Complexity and Real Computation*. Springer (1998)
58. Blum, L., Shub, M., Smale, S.: 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 (1989)
59. Boker, U., Dershowitz, N.: The Church-Turing thesis over arbitrary domains. In: A. Avron, N. Dershowitz, A. Rabinovich (eds.) *Pillars of Computer Science: Essays Dedicated to Boris (Boaz) Trakhtenbrot on the Occasion of His 85th Birthday*, vol. 4800, pp. 199–229. Springer (2008)
60. Boker, U., Dershowitz, N.: Three paths to effectiveness. In: *Fields of Logic and Computation*, pp. 135–146. Springer (2010)
61. Bonnet, J., Yin, P., Ortiz, M.E., Subsoontorn, P., Endy, D.: Amplifying genetic logic gates. *Science* **340**(6132), 599–603 (2013)
62. Bournez, O.: Achilles and the Tortoise climbing up the hyper-arithmetical hierarchy. *Theoretical Computer Science* **210**(1), 21–71 (1999)
63. Bournez, O.: *Complexité algorithmique des systèmes dynamiques continus et hybrides*. Thèse de doctorat, École Normale Supérieure de Lyon (1999)
64. Bournez, O.: How much can analog and hybrid systems be proved (super-)Turing. *Applied Mathematics and Computation* **178**(1), 58–71 (2006)
65. Bournez, O., Campagnolo, M.L.: New Computational Paradigms. *Changing Conceptions of What is Computable*, chap. A Survey on Continuous Time Computations, pp. 383–423. Springer-Verlag, New York (2008)

66. Bournez, O., Campagnolo, M.L., Graça, D., S. Hainry, E.: Polynomial differential equations compute all real computable functions on computable compact intervals. *Journal of Complexity* **23**(3), 317–335 (2007)
67. Bournez, O., Cohen, J., Rabie, M.: Homonym population protocols. In: Springer (ed.) *Networked Systems. Third International Conference, NETYS 2015, Agadir, Morocco, May 13–15, 2015, Revised Selected Papers, Lecture Notes in Computer Science*, vol. 9466 (2015)
68. Bournez, O., Cohen, J., Rabie, M.: Homonym population protocols. *Theory of Computing Systems* **62**(5), 1318–1346 (2018), URL <https://doi.org/10.1007/s00224-017-9833-2>
69. Bournez, O., Dershowitz, N., Falkovich, E.: Towards an axiomatization of simple analog algorithms. In: M. Agrawal, S.B. Cooper, A. Li (eds.) *Theory and Applications of Models of Computation - 9th Annual Conference, TAMC 2012, Beijing, China, May 16–21, 2012. Proceedings, Lecture Notes in Computer Science*, vol. 7287, pp. 525–536. Springer-Verlag (2012)
70. Bournez, O., Dershowitz, N., Néron, P.: Axiomatizing analog algorithms. In: A. Beckmann, L. Bienvenu, N. Jonoska (eds.) *Pursuit of the Universal, Lecture Notes in Computer Science*, vol. 9709, pp. 215–224. Springer, Switzerland (2016). 12th Conference on Computability in Europe, CiE 2016, Paris, France, June 27 - July 1, 2016
71. Bournez, O., Fraigniaud, P., Koegler, X.: Computing with large populations using interactions. In: B. Rovan, V. Sassone, P. Widmayer (eds.) *Mathematical Foundations of Computer Science, MFCS'12, Lecture Notes in Computer Science*. Springer-Verlag (2012)
72. Bournez, O., Graça, D., Pouly, A.: On the functions generated by the general purpose analog computer. Tech. rep. (2016). Under review for *Information and Computation* (current status: accepted for publication under minor revision)
73. Bournez, O., Graça, D.S., Pouly, A.: Polynomial time corresponds to solutions of polynomial ordinary differential equations of polynomial length. *Journal of the ACM* (2017). Accepted for publication
74. Bournez, O., Graça, D.S., Pouly, A.: Polynomial time corresponds to solutions of polynomial ordinary differential equations of polynomial length. the general purpose analog computer and computable analysis are two efficiently equivalent models of computations. In: 43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11–15, 2016, Rome, Italy, *LIPICs*, vol. 55, pp. 109:1–109:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2016)
75. Bournez, O., Hainry, E.: Elementarily computable functions over the real numbers and r -sub-recursive functions. *Theoretical Computer Science* **348**, 130–147 (2005)
76. Bournez, O., Hainry, E.: Recursive analysis characterized as a class of real recursive functions. *Fundamenta Informaticae* **74**(4), 409–433 (2006)
77. Bournez, O., Pouly, A.: A universal ordinary differential equation. In: International Colloquium on Automata Language Programming, ICALP'2017 (2017)
78. Bouyer, P., Fahrenberg, U., Larsen, K.G., Markey, N., Ouaknine, J., Worrell, J.: Model checking real-time systems. In: E. Clarke, T. Henzinger, H. Veith (eds.) *Handbook of Model Checking*. Springer (2017). To appear
79. Bouyer, P., Petit, A.: Decomposition and composition of timed automata. In: J. Wiedermann, P. van Emde Boas, M. Nielsen (eds.) *Automata, Languages and Programming, 26th International Colloquium, ICALP'99, Prague, Czech Republic, July 11–15, 1999, Proceedings, Lecture Notes in Computer Science*, vol. 1644, pp. 210–219. Springer (1999)
80. Bouyer, P., Petit, A.: A Kleene/Büchi-like theorem for clock languages. *Journal of Automata, Languages and Combinatorics* **7**(2), 167–186 (2002)
81. Bowles, M.D.: U.S. technological enthusiasm and British technological skepticism in the age of the analog brain **18**(4), 5–15 (1996)
82. Bozapalidis, S.: Extending stochastic and quantum functions. *Theory of computing systems* **36**(2), 183–197 (2003)
83. Brattka, V.: Computable selection in analysis. In: K.I. Ko, K. Weihrauch (eds.) *Computability and Complexity in Analysis, Informatik Berichte*, vol. 190, pp. 125–138. FernUniversität Hagen (1995). CCA Workshop, Hagen, August 19–20, 1995

84. Braverman, M., Grigo, A., Rojas, C.: Noise vs computational intractability in dynamics. In: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, pp. 128–141. ACM (2012)
85. Braverman, M., Rojas, C., Schneider, J.: Tight space-noise tradeoffs in computing the ergodic measure. arXiv preprint arXiv:1508.05372 (2015)
86. Braverman, M., Schneider, J., Rojas, C.: Space-bounded Church-Turing thesis and computational tractability of closed systems. *Physical review letters* **115**(9), 098,701 (2015)
87. Brihaye, T., Doyen, L., Geeraerts, G., Ouaknine, J., Raskin, J.F., Worrell, J.: On reachability for hybrid automata over bounded time. In: International Colloquium on Automata, Languages, and Programming, pp. 416–427. Springer (2011)
88. Brijder, R., Doty, D., Soloveichik, D.: Robustness of expressivity in chemical reaction networks. In: International Conference on DNA-Based Computers, pp. 52–66. Springer (2016)
89. Brockett, R.W.: Smooth dynamical systems which realize arithmetical and logical operations. In: H. Nijmeijer, J.M. Schumacher (eds.) *Three Decades of Mathematical Systems Theory, Lecture Notes in Computer Science*, vol. 135, pp. 19–30. Springer (1989)
90. Brockett, R.W.: Dynamical systems that sort lists, diagonalize matrices, and solve linear programming problems. *Linear Algebra and its Applications* **146**, 79–91 (1991)
91. Bush, V.: The differential analyser. *Journal of the Franklin Institute* **212**(4), 447–488 (1931)
92. Cabessa, J.: Part I computational capabilities of recurrent neural networks. part II limit knowledge: A topological approach to epistemic game theory. Ph.D. thesis, Université Panthéon-Assas – Paris II Laboratoire d’Économie Mathématique et de Microéconomie Appliquée (LEMMA) (2016)
93. Cabessa, J., Siegelmann, H.T.: Evolving recurrent neural networks are super-Turing. In: Neural Networks (IJCNN), The 2011 International Joint Conference on, pp. 3200–3206. IEEE (2011)
94. Cabessa, J., Siegelmann, H.T.: The computational power of interactive recurrent neural networks. *Neural Computation* **24**(4), 996–1019 (2012)
95. Cabessa, J., Siegelmann, H.T.: The super-Turing computational power of plastic recurrent neural networks. *International journal of neural systems* **24**(08), 1450,029 (2014)
96. Cabessa, J., Villa, A.E.: The expressive power of analog recurrent neural networks on infinite input streams. *Theoretical Computer Science* **436**, 23–34 (2012)
97. Cabessa, J., Villa, A.E.: The super-Turing computational power of interactive evolving recurrent neural networks. In: International Conference on Artificial Neural Networks, pp. 58–65. Springer (2013)
98. Cabessa, J., Villa, A.E.: An attractor-based complexity measurement for boolean recurrent neural networks. *PLoS One* **9**(4), e94,204 (2014)
99. Cabessa, J., Villa, A.E.: Computational capabilities of recurrent neural networks based on their attractor dynamics. In: Neural Networks (IJCNN), 2015 International Joint Conference on, pp. 1–8. IEEE (2015)
100. Cabessa, J., Villa, A.E.: Expressive power of first-order recurrent neural networks determined by their attractor dynamics. *Journal of Computer and System Sciences* **82**(8), 1232–1250 (2016)
101. Calude, C.S., Pavlov, B.: Coins, quantum measurements, and Turing’s barrier. *Quantum Information Processing* **1**(1-2), 107–127 (2002)
102. Campagnolo, M.L.: Computational complexity of real valued recursive functions and analog circuits. Ph.D. thesis, IST, Universidade Técnica de Lisboa (2001)
103. Campagnolo, M.L.: The complexity of real recursive functions. In: C. Calude, M. Dinneen, F. Peper (eds.) *Unconventional Models of Computation, UMC’02*, no. 2509 in *Lecture Notes in Computer Science*, pp. 1–14. Springer (2002)
104. Campagnolo, M.L.: Continuous time computation with restricted integration capabilities. *Theoretical Computer Science* **317**(4), 147–165 (2004)
105. Campagnolo, M.L., Moore, C., Costa, J.F.: Iteration, inequalities, and differentiability in analog computers. *Journal of Complexity* **16**(4), 642–660 (2001)
106. Campagnolo, M.L., Moore, C., Costa, J.F.: An analog characterization of the Grzegorzczak hierarchy. *Journal of Complexity* **18**(4), 977–1000 (2002)

107. Campagnolo, M.L., Ojakian, K.: The elementary computable functions over the real numbers: applying two new techniques. *Archive for Mathematical Logic* **46**(7–8), 593–627 (2008)
108. Cardelli, L.: Strand algebras for DNA computing. *DNA computing and molecular programming* pp. 12–24 (2009)
109. Cardelli, L., Zavattaro, L.: Turing universality of the biochemical ground form. *Mathematical Structures in Computer Science* **20**(1), 45–73 (2010)
110. Care, C.: Technology for modelling: electrical analogies, engineering practice, and the development of analogue computing. Springer Science & Business Media (2010)
111. Carmantini, G.S., Graben, P.B., Desroches, M., Rodrigues, S.: Turing computation with recurrent artificial neural networks. In: *Proceedings of the 2015th International Conference on Cognitive Computation: Integrating Neural and Symbolic Approaches-Volume 1583*, pp. 1–9. CEUR-WS. org (2015)
112. Cervelle, J.: Constructing continuous systems from discrete cellular automata. In: *Conference on Computability in Europe*, pp. 55–64. Springer (2013)
113. Chaitin, G.: The halting probability omega: Irreducible complexity in pure mathematics. *Milan Journal of Mathematics* **75**(1), 291–304 (2007)
114. Chatzigiannakis, I., Michail, O., Nikolaou, S., Pavlogiannis, A., Spirakis, P.: Passively mobile communicating machines that use restricted space. *Theoretical Computer Science* (2011)
115. Chazelle, B.: Natural algorithms. In: *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 422–431. Society for Industrial and Applied Mathematics (2009)
116. Chazelle, B.: The convergence of bird flocking. *Journal of the ACM (JACM)* **61**(4), 21 (2014)
117. Chazelle, B.: An algorithmic approach to collective behavior. *Journal of Statistical Physics* **158**(3) (2015)
118. Chazelle, B.: Diffusive influence systems. *SIAM Journal on Computing* **44**(5), 1403–1442 (2015)
119. Chen, H.L., Doty, D., Soloveichik, D.: Deterministic function computation with chemical reaction networks. *Natural computing* **13**(4), 517–534 (2014)
120. Chen, H.L., Doty, D., Soloveichik, D.: Rate-independent computation in continuous chemical reaction networks. In: *Proceedings of the 5th conference on Innovations in theoretical computer science*, pp. 313–326. ACM (2014)
121. Chen, Y.J., Dalchau, N., Srinivas, N., Phillips, A., Cardelli, L., Soloveichik, D., Seelig, G.: Programmable chemical controllers made from DNA. *Nature nanotechnology* **8**(10), 755–762 (2013)
122. Chonev, V., Ouaknine, J., Worrell, J.: On the Skolem problem for continuous linear dynamical systems. *arXiv preprint arXiv:1506.00695* (2015)
123. Chonev, V., Ouaknine, J., Worrell, J.: On recurrent reachability for continuous linear dynamical systems. In: *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, pp. 515–524. ACM (2016)
124. Church, G.M., Gao, Y., Kosuri, S.: Next-generation digital information storage in DNA. *Science* **337**(6102), 1628–1628 (2012)
125. Cockshott, P., Mackenzie, L., Michaelson, G.: Physical constraints on hypercomputation. *Theoretical Computer Science* **394**(3), 159–174 (2008)
126. Collins, P.: Continuity and computability on reachable sets. *Theoretical Computer Science* **341**, 162–195 (2005)
127. Collins, P., Lygeros, J.: 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*, pp. 4688–4693. IEEE Computer Society Press (2005)
128. Collins, P., van Schuppen, J.H.: Observability of hybrid systems and Turing machines. In: *43rd IEEE Conference on Decision and Control* (2004)
129. Conrad, M.: Molecular computing. *Advances in Computers* **31**, 235–324 (1990)
130. Cook, M., Soloveichik, D., Winfree, E., Bruck, J.: Programmability of chemical reaction networks. In: *Algorithmic Bioprocesses*, pp. 543–584. Springer (2009)

131. Copeland, B.J.: Even Turing machines can compute uncomputable functions. In: C. Calude, J. Casti, M. Dinneen (eds.) *Unconventional Models of Computations*. Springer-Verlag (1998)
132. Copeland, B.J.: Accelerating Turing machines. *Minds and Machines* **12**, 281–301 (2002)
133. Copeland, B.J.: The Church-Turing thesis. In: E.N. Zalta (ed.) *The Stanford Encyclopedia of Philosophy*. Stanford University (Fall 2002)
134. Copeland, B.J.B.J.: Hypercomputation: philosophical issues. *Theoretical Computer Science* **317**(1–3), 251–267 (2004)
135. Costa, J.F., Leong, R.: The ARNN model relativises $P=NP$ and $P!=NP$. *Theoretical Computer Science* **499**, 2–22 (2013), URL <https://doi.org/10.1016/j.tcs.2013.05.022>
136. Costa, J.F., Loff, B., Mycka, J.: The new promise of analog computation. In: S.B. Cooper, B. Löwe, A. Sorbi (eds.) *Computation and Logic in the Real World, Lecture Notes in Computer Science*, vol. 4497, pp. 189–195. Springer, Berlin (2007). Third Conference on Computability in Europe, CiE 2007, Siena, Italy, June 18–23, 2007
137. Costa, J.F., Loff, B., Mycka, J.: A foundation for real recursive function theory. *Annals of Pure and Applied Logic* **160**(3), 255–288 (2009)
138. Costa, N., Doria, F.: Undecidable Hopf bifurcation with undecidable fixed point. *International Journal of Theoretical Physics* **33**(9), 1885–1903 (1994)
139. Costa, N., Doria, F., Amaral, A.: Dynamical system where proving chaos is equivalent to proving Fermat’s conjecture. *International journal of theoretical physics* **32**(11), 2187–2206 (1993)
140. Couturier, E., Jacquet, N.: [Construction of a universal ordinary differential equation \$|\infty\$ of order 3](#). arXiv preprint arXiv:1610.09148 (2016)
141. Cummings, R., Doty, D., Soloveichik, D.: Probability 1 computation with chemical reaction networks. In: *International Workshop on DNA-Based Computers*, pp. 37–52. Springer (2014)
142. Czyżowicz, J., Gasieniec, L., Kosowski, A., Kranakis, E., Spirakis, P.G., Uznański, P.: On convergence and threshold properties of discrete lotka-volterra population protocols. In: *International Colloquium on Automata, Languages, and Programming*, pp. 393–405. Springer (2015)
143. Davies, E.B.: Building infinite machines. *The British Journal for the Philosophy of Science* **52**, 671–682 (2001)
144. Davis, M.: The myth of hypercomputation. In: *Alan Turing: Life and legacy of a great thinker*, pp. 195–211. Springer (2004)
145. Delporte-Gallet, C., Fauconnier, H., Guerraoui, R., Ruppert, E.: When birds die: Making population protocols fault-tolerant. In: P.B. Gibbons, T.F. Abdelzaher, J. Aspnes, R. Rao (eds.) *Distributed Computing in Sensor Systems, Second IEEE International Conference, DCOSS 2006, San Francisco, CA, USA, June 18–20, 2006, Proceedings, Lecture Notes in Computer Science*, vol. 4026, pp. 51–66. Springer (2006), URL http://dx.doi.org/10.1007/11776178_4
146. Delvenne, J.C., Kurka, P., Blondel, V.D.: Computational universality in symbolic dynamical systems. In: M. Margenstern (ed.) *MCU: International Conference on Machines, Computations, and Universality, Lecture Notes in Computer Science*, vol. 3354, pp. 104–115. Springer (2004)
147. Dershowitz, N., Gurevich, Y.: A natural axiomatization of computability and proof of Church’s thesis. *The Bulletin of Symbolic Logic* **14**(3), 299–350 (2008)
148. Deutsch, D.: Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society (London), Series A* **400**, 97–117 (1985)
149. Diehl, P.U., Cook, M.: Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in computational neuroscience* **9**, 99 (2015)
150. Doty, D.: Timing in chemical reaction networks. In: *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pp. 772–784. Society for Industrial and Applied Mathematics (2014)
151. Dowek, G.: *Les métamorphoses du calcul : une étonnante histoire de mathématiques*. Le Pommier, Paris (2007), URL <http://opac.inria.fr/record=b1122726>

152. Dowek, G.: The physical Church thesis as an explanation of the galileo thesis. *Natural Computing* **11**(2), 247–251 (2012)
153. Durand-Lose, J.: Abstract geometrical computation: Turing-computing ability and undecidability. In: S.B. Cooper, B. Löwe, L. Torenvliet (eds.) *New Computational Paradigms, First Conference on Computability in Europe, CiE 2005, Amsterdam, The Netherlands, June 8–12, 2005, Proceedings, Lecture Notes in Computer Science*, vol. 3526, pp. 106–116. Springer (2005), URL http://dx.doi.org/10.1007/11494645_14
154. Durand-Lose, J.: Abstract geometrical computation I: Embedding black hole computations with rational numbers. *Fundam. Inform.* **74**(4), 491–510 (2006), URL <http://content.iospress.com/articles/fundamenta-informaticae/fi74-4-07>
155. Durand-Lose, J., Jonoska, N. (eds.): *Tractional Motion Machines: Tangent-Managing Planar Mechanisms as Analog Computers and Educational Artifacts, Lecture Notes in Computer Science*, vol. 7445. Springer (2012)
156. Durand-Richard, M.J.: Charles babbage (1791-1871): de l'école algébrique anglaise à la 'machine analytique'. *Mathématiques, informatique et sciences humaines* **118**, 5–31 (1992)
157. Durand-Richard, M.J.: Planimeters and integrals in the 19th century. before the differential analyser. *Nuncius* **25**(1), 101–124 (2010)
158. Durand-Richard, M.J.: Entre Ciel et Mer. Des observatoires pour l'enseignement de l'astronomie, des sciences maritimes et le service de l'heure, en France et en Europe, de la fin du XVIIIe au début du XXe siècle : institutions, pratiques et cultures, chap. De la prédiction des marées : entre calcul, observation et mécanisation (1831-1876). No. 8-9 in *II* (2016)
159. Durand-Richard, M.J.: Douglas R. Hartree (1897-1958): de l'analyseur différentiel à l'ordinateur. *Revue de Synthèse* (2017)
160. Earman, J., Norton, J.D.: Forever is a day: Supertasks in Pitowsky and Malament-Hogarth spacetimes. *Philosophy of Science* **60**(1), 22–42 (1993)
161. Ercsey-Ravasz, M., Toroczkai, Z.: Optimization hardness as transient chaos in an analog approach to constraint satisfaction. *Nature Physics* **7**(12), 966–970 (2011)
162. Esparza, J., Ganty, P., Leroux, J., Majumdar, R.: Verification of population protocols. *Acta Informatica* pp. 1–25 (2016)
163. Etesi, G., Németi, I.: Non-Turing computations via Malament-Hogarth space-times. *International Journal of Theoretical Physics* **41**(2), 341–370 (2002)
164. Evans, C.G., Winfree, E.: Physical principles for DNA tile self-assembly. *Chemical Society Reviews* **46**(12), 3808–3829 (2017)
165. Fages, F.: Cells as machines: towards deciphering biochemical programs in the cell. In: *International Conference on Distributed Computing and Internet Technology*, pp. 50–67. Springer (2014)
166. Fages, F., Le Guludec, G., Bournez, O., Pouly, A.: Strong Turing completeness of continuous chemical reaction networks and compilation of mixed analog-digital programs. In: *Computational Methods in Systems Biology-CMSB 2017* (2017)
167. Fages, F., Soliman, S.: Abstract interpretation and types for systems biology. *Theoretical Computer Science* **403**(1), 52–70 (2008)
168. Faybusovich, L.: Dynamical systems which solve optimization problems with linear constraints. *IMA Journal of Mathematical Control and Information* **8**, 135–149 (1991)
169. Fischer, J.: Instrumente zur mechanischen integration. ein zwischenbericht. *Brückenschläge* **25**, 111–156 (1995)
170. Fischer, J.: Instrumente zur mechanischen integration ii. ein (weiterer) zwischenbericht. *Chemie-Kultur-Geschichte. Festschrift für Hans-Werner Schütt anlässlich seines 65.*, 143–155 (2002)
171. Franchette, F.: La thèse de l'hyper-calcul : enjeux et problèmes philosophiques. *Philosophia Scientiæ* **16**(3) (2012)
172. Francisco, A.P.L.: Finite automata over continuous time (2002). Diploma Thesis. Universidade Técnica de Lisboa, Instituto Superior Técnico
173. Fredkin, E., Toffoli, T.: Conservative logic. In: *Collision-based computing*, pp. 47–81. Springer (2002)

174. Friedman, H.: Algorithmic procedures, generalized Turing algorithms, and elementary recursion theory. In: *Studies in Logic and the Foundations of Mathematics*, vol. 61, pp. 361–389. Elsevier (1971)
175. Gandhi, A., Khossainov, B., Liu, J.: Finite automata over structures. In: *TAMC*, pp. 373–384. Springer (2012)
176. Gandy, R.: Church’s thesis and principles for mechanisms. *The Kleene Symposium* pp. 123–148 (1980)
177. Gaßner, C.: The separation of relativized versions of P and DNP for the ring of the reals. *J. UCS* **16**(18), 2563–2568 (2010)
178. Gaßner, C.: Strong Turing degrees for additive BSS ram’s. *Logical Methods in Computer Science* (2013)
179. Ghosh-Dastidar, S., Adeli, H.: Spiking neural networks. *International journal of neural systems* **19**(04), 295–308 (2009)
180. Gill, P.E., Murray, W., Saunders, M.A., Tomlin, J.A., Wright, M.H.: On projected Newton barrier methods for linear programming and an equivalence to karmarkar’s projective method. *Mathematical programming* **36**(2), 183–209 (1986)
181. Gillespie, D.: A rigorous derivation of the chemical master equation. *Physica A* **188**(1-3), 404–425 (1992)
182. Gillespie, D.T.: Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.* **81**(25), 2340–2361 (1977)
183. Gladkikh, A., Malinetskii, G.: Study of dynamical systems from the viewpoint of complexity and computational capabilities. *Differential Equations* **52**(7), 897–905 (2016)
184. Gori, M., Meer, K.: A step towards a complexity theory for analog systems. *Mathematical Logic Quarterly* **48**(S1), 45–58 (2002)
185. Gori, M., Meer, K.: Some aspects of a complexity theory for continuous time systems. In: *Conference on Computability in Europe*, pp. 554–565. Springer (2007)
186. Graça, D.S.: Some recent developments on Shannon’s general purpose analog computer. *Mathematical Logic Quarterly* **50**(4,5), 473–485 (2004)
187. Graça, D.S., Campagnolo, M.L., Buescu, J.: Robust simulations of Turing machines with analytic maps and flows. In: B. Cooper, B. Loewe, L. Torenvliet (eds.) *Proceedings of CiE’05, New Computational Paradigms, Lecture Notes in Computer Science*, vol. 3526, pp. 169–179. Springer-Verlag (2005)
188. Graça, D.S., Costa, J.F.: Analog computers and recursive functions over the reals. *Journal of Complexity* **19**(5), 644–664 (2003)
189. Graves, A., Wayne, G., Danihelka, I.: Neural Turing machines. *arXiv preprint arXiv:1410.5401* (2014)
190. Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., Colmenarejo, S.G., Grefenstette, E., Ramalho, T., Agapiou, J., et al.: Hybrid computing using a neural network with dynamic external memory. *Nature* **538**(7626), 471–476 (2016)
191. Grefenstette, E., Hermann, K.M., Suleyman, M., Blunsom, P.: Learning to transduce with unbounded memory. In: *Advances in Neural Information Processing Systems*, pp. 1828–1836 (2015)
192. Gruska, J.: *Foundations of Computing*. International Thomson Publishing (1997)
193. Gu, J., Gu, Q., Du, D.: On optimizing the satisfiability (SAT) problem. *Journal of Computer Science and Technology* **14**(1), 1–17 (1999)
194. Guerraoui, R., Ruppert, E.: Names trump malice: Tiny mobile agents can tolerate byzantine failures. *Automata, Languages and Programming* pp. 484–495 (2009)
195. Gupta, I., Nagda, M., Devaraj, C.F.: The design of novel distributed protocols from differential equations. *Distributed Computing* **20**(2), 95–114 (2007)
196. Gurevich: Sequential abstract-state machines capture sequential algorithms. *ACMTCL: ACM Transactions on Computational Logic* **1** (2000)
197. Haigh, T., Priestley, P.M., Priestley, M., Rope, C.: *ENIAC in action: Making and remaking the modern computer*. MIT press (2016)
198. Hartmann, L., Jones, N.D., Simonsen, J.G.: Programming in biomolecular computation. *Electronic Notes in Theoretical Computer Science* **268**, 97–114 (2010)

199. Hashagen, U.: Rechner für die wissenschaft: "scientific computing" und informatik im deutschen wissenschaftssystem 1870-1970. *Rechnende Maschinen im Wandel: Mathematik, Technik, Gesellschaft*. Deutsches Museum pp. 111–152 (2011)
200. Hashagen, U.: Analog computing as a failed modernization program in germany 1930-1960 (2017). Talk at "Histoire et Philosophie de l'Informatique", MESHS, Lille
201. Head, T.: Formal language theory and DNA: An analysis of the generative capacity of specific recombinant behaviors. *Bulletin of Mathematical Biology* **49**, 737–759 (1987)
202. Helmke, U., Moore, J.: *Optimization and Dynamical Systems*. Communications and Control Engineering Series. Springer Verlag, London (1994)
203. Helmke, U., Moore, J.B.: *Optimization and dynamical systems*. Springer Science & Business Media (2012)
204. Henzinger, T.A., Kopke, P.W., Puri, A., Varaiya, P.: What's decidable about hybrid automata? *Journal of Computer and System Sciences* **57**(1), 94–124 (1998)
205. Hiebeler, D.: Dynamics and resistance to neighborhood perturbations of discrete-and continuous-time cellular automata. *J. Cellular Automata* **1**(2), 125–139 (2006)
206. Hochreiter, S., Schmidhuber, J.: Long short-term memory. *Neural computation* **9**(8), 1735–1780 (1997)
207. Hofbauer, J., Sigmund, K.: Evolutionary game dynamics. *Bulletin of the American Mathematical Society* **4**, 479–519 (2003)
208. Hogarth, M.: Non-Turing computers and non-Turing computability. In: *Proceedings of the Philosophy of Science Association (PSA'94)*, vol. 1, pp. 126–138 (1994)
209. Hogarth, M.: Predictability, computability and spacetime. Ph.D. thesis, Sidney Sussex College, Cambridge (1996)
210. Hogarth, M.: Non-Turing computers are the new non-Euclidean geometries. In: *Future Trends in Hypercomputation* (2006). Sheffield, 11–13 September 2006. Available for download on www.hypercomputation.net
211. Hogarth, M.L.: Does general relativity allow an observer to view an eternity in a finite time? *Foundations of Physics Letters* **5**, 173–181 (1992)
212. Hopfield, J.J.: 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 (1984)
213. Huang, X., Klinge, T.H., Lathrop, J.I., Li, X., Lutz, J.H.: Real-time computability of real numbers by chemical reaction networks. In: *International Conference on Unconventional Computation and Natural Computation*, pp. 29–40. Springer (2017)
214. Jeandel, E.: Topological automata. *Theory of Computing Systems* **40**(4), 397–407 (2007)
215. Karmarkar, N.: A new polynomial-time algorithm for linear programming. In: *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pp. 302–311. ACM (1984)
216. Kempe, A.: On a general method of describing plane curves of the n -th degree by linkwork. *Proceedings of the London Mathematical Society* **7**, 213–216 (1876)
217. Khachiyan, L.G.: Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics* **20**(1), 53–72 (1980)
218. Kieu, T.D.: Hypercomputation with quantum adiabatic processes. *Theoretical Computer Science* **317**(1-3), 93–104 (2004)
219. Koiran, P.: The topological entropy of iterated piecewise affine maps is uncomputable. *Discrete Mathematics & Theoretical Computer Science* **4**(2), 351–356 (2001)
220. Koiran, P., Cosnard, M., Garzon, M.: Computability with low-dimensional dynamical systems. *Theoret. Comput. Sci.* **132**(1-2), 113–128 (1994)
221. Koiran, P., Moore, C.: Closed-form analytic maps in one and two dimensions can simulate universal Turing machines. *Theoretical Computer Science* **219**, 217–223 (1999)
222. Krivine, H., Lesne, A., Treiner, J.: Discrete-time and continuous-time modeling: some bridges and gaps. *Mathematical Structures in Computer Science* (2006). In print
223. Kurgansky, O., Potapov, I.: Computation in one-dimensional piecewise maps and planar pseudo-billiard systems. In: C. Calude, M.J. Dinneen, G. Paun, M.J. Pérez-Jiménez, G. Rozenberg (eds.) *Unconventional Computation, 4th International Conference, UC 2005*, Sevilla, Spain, October 3-7, 2005, *Proceedings, Lecture Notes in Computer Science*, vol. 3699, pp. 169–175. Springer (2005)

224. Kurtz, T.: Approximation of population processes, volume 36 of cbmsnsf regional conf. Series in Appl. Math. SIAM, Philadelphia (1981)
225. Lanotte, R., Tini, S.: Taylor approximation for hybrid systems. *Information and Computation* **205**(11), 1575–1607 (2007)
226. Lee, E.A.: Models of time for CPS. Course at UC Berkeley (available online) (2015)
227. Legenstein, R., Maass, W.: What makes a dynamical system computationally powerful? In: S. Haykin, J.C. Principe, T. Sejnowski, J. McWhirter (eds.) *New Directions in Statistical Signal Processing: From Systems to Brain*, pp. 127–154. MIT Press (2007)
228. Lesne, A.: Discrete vs continuous controversy in physics. *Mathematical Structures in Computer Science* (2006). In print
229. Lipshitz, L., Rubel, L.A.: A differentially algebraic replacement theorem, and analog computability. *Proceedings of the American Mathematical Society* **99**(2), 367–372 (1987)
230. Lipton, R.J.: DNA solution of hard computational problems. *Science* **268**, 542–545 (1995)
231. Loff, B.: A functional characterisation of the analytical hierarchy. In: *Computability in Europe 2007: Computation and Logic in the Real World*. (2007)
232. Loff, B., Costa, J.F., Mycka, J.: Computability on reals, infinite limits and differential equations. *Applied Mathematics and Computation* **191**(2), 353–391 (2007). To appear
233. Loff, B., Costa, J.F., Mycka, J.: The new promise of analog computation. In: *Computability in Europe 2007: Computation and Logic in the Real World*. (2007)
234. Longo, G., Paul, T.: The mathematics of computing between logic and physics. *Computability in Context: Computation and Logic in the Real World*; Cooper, S., Sorbi, A., Eds pp. 243–274 (2011)
235. Lovgren, S.: Computer made from DNA and enzymes. *National Geographic News* **24** (2003)
236. Maass, W., Bishop, C.: *Pulsed Neural Networks*. Cambridge MA. MIT Press (1998)
237. Maass, W., Orponen, P.: On the effect of analog noise in discrete-time analog computations. *Neural Computation* **10**(5), 1071–1095 (1998)
238. Maass, W., Sontag, E.: Analog neural nets with gaussian or other common noise distributions cannot recognize arbitrary regular languages. *Neural Computation* **11**(3), 771–782 (1999)
239. MacLennan, B.: *Field computation: A theoretical framework for massively parallel analog computation. parts i-iv*. University of Tennessee, Computer Science Department (1990)
240. MacLennan, B.: The promise of analog computation. *International Journal of General Systems* **43**(7), 682–696 (2014)
241. MacLennan, B.J.: Field computation in natural and artificial intelligence. *Information Sciences* **119**(1), 73–89 (1999)
242. MacLennan, B.J.: Natural computation and non-turing models of computation. *Theoretical computer science* **317**(1), 115–145 (2004)
243. MacLennan, B.J.: Analog computation. In: *Encyclopedia of complexity and systems science*, pp. 271–294. Springer (2009)
244. Margolus, N.: Physics-like models of computation. *Physica D: Nonlinear Phenomena* **10**(1-2), 81–95 (1984)
245. Mather, J., McGehee, R.: Solutions of the collinear four body problem which become unbounded in finite time. In: *Dynamical systems, theory and applications*, pp. 573–597. Springer (1975)
246. Meer, K.: On ladner’s result for a class of real machines with restricted use of constants. *Information and Computation* **210**, 13–20 (2012)
247. Meer, K., Naif, A.: [Generalized finite automata over real and complex numbers](#). *Theoretical Computer Science* **591**, 85–98 (2015)
248. Meer, K., Naif, A.: Periodic generalized automata over the reals. In: *International Conference on Language and Automata Theory and Applications*, pp. 168–180. Springer (2016)
249. Merolla, P.A., Arthur, J.V., Alvarez-Icaza, R., Cassidy, A.S., Sawada, J., Akopyan, F., Jackson, B.L., Imam, N., Guo, C., Nakamura, Y., et al.: A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science* **345**(6197), 668–673 (2014)
250. Messick, S.: Compactness in the theory of continuous automata. In: *International Symposium on Logical Foundations of Computer Science*, pp. 251–266. Springer (2016)

251. Michail, O., Chatzigiannakis, I., Spirakis, P.G.: New models for population protocols. *Synthesis Lectures on Distributed Computing Theory* **2**(1), 1–156 (2011)
252. Mickelson, N.B.: Analog computing device (1963). US Patent 3,113,170
253. Milici, P.: Tractional motion machines extend gpac-generable functions. *IJUC* **8**(3), 221–233 (2012), URL <http://www.oldcitypublishing.com/IJUC/IJUCabstracts/IJUC8.3abstracts/IJUCv8n3p221-233Milici>
254. Mills, J.: Programmable VLSI extended analog computer for cyclotron beam control. Tech. Rep. 441, Indiana University Computer Science (1995)
255. Mills, J.W.: The nature of the extended analog computer. *Physica D: Nonlinear Phenomena* **237**(9), 1235 – 1256 (2008). Novel Computing Paradigms: Quo Vadis?
256. Mills, J.W., Himebaugh, B., Allred, A., Bulwinkle, D., Deckard, N., Gopalakrishnan, N., Miller, J., Miller, T., Nagai, K., Nakamura, J., Ololowe, B., Vlas, R., Whitener, P., Ye, M., , Zhang, C.: 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 (2005)
257. Moore, C.: Unpredictability and undecidability in dynamical systems. *Physical Review Letters* **64**(20), 2354–2357 (1990)
258. Moore, C.: Recursion theory on the reals and continuous-time computation. *Theoretical Computer Science* **162**(1), 23–44 (1996)
259. Moore, C., Mertens, S.: The nature of computation. OUP Oxford (2011)
260. Moschovakis, Y.N.: Abstract recursion as a foundation for the theory of algorithms. In: *Computation and Proof Theory, Lecture Notes in Mathematics*, vol. 1104, pp. 289–364. Springer-Verlag, Berlin (1984)
261. Mycka, J., Coelho, F., Costa, J.F.: The euclid abstract machine: Trisection of the angle and the halting problem. In: C.S. Calude, M.J. Dinneen, G. Paun, G. Rozenberg, S. Stepney (eds.) *Unconventional Computation, 5th International Conference, UC 2006, York, UK, September 4-8, 2006, Proceedings, Lecture Notes in Computer Science*, vol. 4135, pp. 195–206. Springer (2006), URL http://dx.doi.org/10.1007/11839132_16
262. Mycka, J., Costa, J.F.: Real recursive functions and their hierarchy. *Journal of Complexity* **20**(6), 835–857 (2004)
263. Mycka, J., Costa, J.F.: What lies beyond the mountains? Computational systems beyond the Turing limit. *European Association for Theoretical Computer Science Bulletin* **85**, 181–189 (2005)
264. Mycka, J., Costa, J.F.: The $P \neq NP$ conjecture in the context of real and complex analysis. *Journal of Complexity* **22**(2), 287–303 (2006)
265. Nagamatu, M., Yanaru, T.: On the stability of lagrange programming neural networks for satisfiability problems of prepositional calculus. *Neurocomputing* **13**(2), 119–133 (1996)
266. Neher, M.: Interval methods and taylor model methods for odes. TM VII, Key West, KIT, Michigan State University p. 33 (2011)
267. Némethi, I., Andr  ka, H.: New physics and hypercomputation. In: J. Wiedermann, G. Tel, J. Pokorn  y, M. Bielikov  , J. Stuller (eds.) *SOFSEM 2006: Theory and Practice of Computer Science, 32nd Conference on Current Trends in Theory and Practice of Computer Science, Mer  n, Czech Republic, January 21-27, 2006, Proceedings, Lecture Notes in Computer Science*, vol. 3831, p. 63. Springer (2006)
268. N  methi, I., Gyula, D.: Relativistic computers and the Turing barrier. *Applied Mathematics and Computation* **178**(1), 118–142 (2006)
269. Orponen, P.: A survey of continuous-time computation theory. In: D.Z. Du, K.I. Ko (eds.) *Advances in Algorithms, Languages, and Complexity*, pp. 209–224. Kluwer Academic Publishers (1997)
270. Orponen, P., Matamala, M.: Universal computation by finite two-dimensional coupled map lattices. In: *Proceedings, physics and computation*, pp. 243–7. New England Complex Systems Institute Cambridge, MA (1996)
271. Owens, L.: Vannevar bush and the differential analyzer: the text and context of an early computer. *Technology and culture* **27**(1), 63–95 (1986)

272. Păun, G.: Membrane Computing. An Introduction. Springer-Verlag, Berlin (2002)
273. Paun, G., Rozenberg, G.: A guide to membrane computing. *Theoretical Computer Science* **287**(1), 73–100 (2002)
274. Pégny, M.: Les deux formes de la thèse de Church-Turing et l'épistémologie du calcul. *Philosophia Scientiæ* **16**(3) (2012)
275. Pégny, M.: Computational complexity: An empirical view. In: 39th Annual Convention of the Society for the Study of Artificial Intelligence and the Simulation of Behaviour (AISB 2013), Proceedings of the 2013 AISB Convention. Exeter, United Kingdom (2013)
276. Pégny, M.: Sur les limites empiriques du calcul: calculabilité, complexité et physique. Ph.D. thesis, Paris 1 (2013)
277. Pégny, M.: How to make a meaningful comparison of models: The Church–Turing thesis over the reals. *Minds and Machines* **26**(4), 359–388 (2016)
278. Platzer, A.: Differential dynamic logic for hybrid systems. *J. Autom. Reas.* **41**(2), 143–189 (2008)
279. Platzer, A.: Logical analysis of hybrid systems: proving theorems for complex dynamics. Springer Science & Business Media (2010)
280. Platzer, A.: The complete proof theory of hybrid systems. In: Proceedings of the 2012 27th Annual IEEE/ACM Symposium on Logic in Computer Science, pp. 541–550. IEEE Computer Society (2012)
281. Platzer, A.: A uniform substitution calculus for differential dynamic logic. In: International Conference on Automated Deduction, pp. 467–481. Springer (2015)
282. Platzer, A.: Logic & proofs for cyber-physical systems. In: International Joint Conference on Automated Reasoning, pp. 15–21. Springer (2016)
283. Platzer, A.: Logical Foundations of Cyber-Physical Systems. Springer, Switzerland (2018), URL <http://www.springer.com/978-3-319-63587-3>
284. Poças, D., Zucker, J.: Fixed point techniques in analog systems. In: Mathematical and Computational Approaches in Advancing Modern Science and Engineering, pp. 701–711. Springer (2016)
285. Poizat, B.: Les petits cailloux: Une approche modèle-théorique de l'Algorithmie. Aléas Editeur (1995)
286. Pour-El, M.B.: 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 (1974)
287. Qian, L., Soloveichik, D., Winfree, E.: Efficient Turing-universal computation with DNA polymers. In: Proc. DNA Computing and Molecular Programming, *LNCS*, vol. 6518, pp. 123–140. Springer-Verlag (2011)
288. Rabinovich, A.: Automata over continuous time. *Theoretical Computer Science* **300**(1–3), 331–363 (2003)
289. Rabinovich, A.M., Trakhtenbrot, B.A.: From finite automata toward hybrid systems (extended abstract). In: B.S. Chlebus, L. Czaja (eds.) *FCT, Lecture Notes in Computer Science*, vol. 1279, pp. 411–422. Springer (1997)
290. Renegar, J.: A mathematical view of interior-point methods in convex optimization, vol. 3. Siam (2001)
291. Reus, B.: Limits of Computation: From a Programming Perspective, chap. Molecular Computing. Springer (2016)
292. Rojas, R., Hashagen, U.: The first computers: History and architectures. MIT press (2002)
293. Rozenberg, G., Bck, T., Kok, J.N.: Handbook of natural computing. Springer Publishing Company, Incorporated (2011)
294. Rubel, L.A.: A universal differential equation. *Bulletin of the American Mathematical Society* **4**(3), 345–349 (1981)
295. Rubel, L.A.: The extended analog computer. *Advances in Applied Mathematics* **14**, 39–50 (1993)
296. Ruohonen, K.: Undecidability of event detection for ODEs. *Journal of Information Processing and Cybernetics* **29**, 101–113 (1993)

297. Ruohonen, K.: Event detection for ODEs and nonrecursive hierarchies. In: Proceedings of the Colloquium in Honor of Arto Salomaa. Results and Trends in Theoretical Computer Science (Graz, Austria, June 10-11, 1994), *Lecture Notes in Computer Science*, vol. 812, pp. 358–371. Springer-Verlag, Berlin (1994), URL <http://springerlink.metapress.com/openurl.asp?genre=article&issn=0302-9743&volume=812&page=358>
298. Ruohonen, K.: Undecidable event detection problems for ODEs of dimension one and two. *Theoretical Informatics and Applications* **31**(1), 67–79 (1997)
299. Ruohonen, K.: Chomskian hierarchies of families of sets of piecewise continuous functions. *Theory of Computing Systems* **37**(5), 609–638 (2004)
300. Saari, D.G., Xia, Z.: Off to infinity in finite time. *Notices of the AMS* **42**(5) (1995)
301. Scalise, D., Schulman, R.: Designing modular reaction-diffusion programs for complex pattern formation. *Technology* **2**(01), 55–66 (2014)
302. Scalise, D., Schulman, R.: Emulating cellular automata in chemical reaction-diffusion networks. *Natural Computing* **15**(2), 197–214 (2016)
303. Schaller, M., Svozil, K.: Scale-invariant cellular automata and self-similar petri nets. *The European Physical Journal B* **69**(2), 297–311 (2009)
304. Schönhage, A.: Storage modification machines. *SIAM Journal on Computing* **9**(3), 490–508 (1980)
305. Shannon, C.E.: Mathematical theory of the differential analyser. *Journal of Mathematics and Physics MIT* **20**, 337–354 (1941)
306. Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: S. Goldwasser (ed.) Proceedings of the 35th Annual Symposium on Foundations of Computer Science, pp. 124–134. IEEE Computer Society Press, Los Alamitos, CA, USA (1994)
307. Siegelmann, H.T.: Neural Networks and Analog Computation: Beyond the Turing Limit. Birkhäuser (1999)
308. Siegelmann, H.T., Sontag, E.D.: Analog computation via neural networks. *Theoretical Computer Science* **131**(2), 331–360 (1994)
309. Siegelmann, H.T., Sontag, E.D.: On the computational power of neural nets. *Journal of Computer and System Sciences* **50**(1), 132–150 (1995)
310. Šíma, J., Orponen, P.: Continuous-time symmetric Hopfield nets are computationally universal. *Neural Computation* **15**(3), 693–733 (2003)
311. Šíma, J., Orponen, P.: General-purpose computation with neural networks: A survey of complexity theoretic results. *Neural Computation* **15**(12), 2727–2778 (2003)
312. Smith, W.D.: Plane mechanisms and the downhill principle (1998)
313. Smith, W.D.: Church’s thesis meets quantum mechanics. Tech. rep., NEC Research Institute (2004)
314. Smith, W.D.: Church’s thesis meets the N-body problem. *Applied Mathematics and Computation* **178**(1), 154–183 (2006)
315. Soloveichik, D.: The computational power of chemical reaction networks. Banff International Research Station. (2014)
316. Soloveichik, D., Cook, M., Winfree, E., Bruck, J.: Computation with finite stochastic chemical reaction networks. *natural computing* **7**(4), 615–633 (2008)
317. Soloveichik, D., Seelig, G., Winfree, E.: DNA as a universal substrate for chemical kinetics. *Proceedings of the National Academy of Sciences* **107**(12), 5393–5398 (2010)
318. Soloveichik, D., Winfree, E.: Complexity of self-assembled shapes. *SIAM Journal on Computing* **36**(6), 1544–1569 (2007)
319. Stemmer, W.P.: The evolution of molecular computation. *Science* **270**(5241), 1510–1511 (1995)
320. Sun, G., Giles, C., Chen, H., Lee, Y.: The neural network pushdown automation: model, stack and learning simulations. arXiv preprint arXiv:1711.05738 (1993)
321. Syropoulos, A.: Hypercomputation: computing beyond the Church-Turing barrier. Springer Science & Business Media (2008)
322. Tero, A., Kobayashi, R., Nakagaki, T.: A mathematical model for adaptive transport network in path finding by true slime mold. *Journal of theoretical biology* **244**(4), 553–564 (2007)

323. Thomson, W.: On an instrument for calculating the integral of the product of two given functions. In: Proceedings of the Royal Society of London, vol. 24, pp. 266–276 (1876)
324. Trakhtenbrot, B.A.: Automata and their interaction: Definitional suggestions. In: G. Ciobanu, G. Paun (eds.) FCT, *Lecture Notes in Computer Science*, vol. 1684, pp. 54–89. Springer (1999)
325. Tucker, J., Zucker, J.: Computable functions and semicomputable sets on many-sorted algebras. In: S. Abramsky, D. Gabbay, T. Maibaum (eds.) *Handbook of Logic in Computer Science*, Volume 5, pp. 317–523. Oxford University Press, Oxford (2000)
326. Tucker, J.V., Zucker, J.I.: Computability of analog networks. *Theoretical Computer Science* **371**(1–2), 115–146 (2007)
327. Tucker, J.V., Zucker, J.I.: Computability of operators on continuous and discrete time streams. *Computability* **3**(1), 9–44 (2014)
328. Ulmann, B.: *Analog computing*. Walter de Gruyter (2013)
329. Vergis, A., Steiglitz, K., Dickinson, B.: The complexity of analog computation. *Mathematics and Computers in Simulation* **28**(2), 91–113 (1986)
330. Wah, B.W., Chang, Y.J.: Trace-based methods for solving nonlinear global optimization and satisfiability problems. *Journal of Global Optimization* **10**(2), 107–141 (1997)
331. Weibull, J.W.: *Evolutionary Game Theory*. The MIT Press (1995)
332. Weihrauch, K.: *Computable Analysis*. Springer, Berlin (2000)
333. Welch, P.D.: The extent of computation in Malament-Hogarth spacetimes (2006)
334. Whyman, R.: Physical computation, P/poly and P/log. In: PC 2016 (2016)
335. Williams, M.R.: *A history of computing technology*. IEEE Computer Society Press (1997)
336. Woods, D., Naughton, T.J.: An optical model of computation. *Theoretical Computer Science* **334**(1–3), 227–258 (2005)
337. Yin, X., Sedighi, B., Varga, M., Ercsey-Ravasz, M., Toroczkai, Z., Hu, X.S.: Efficient analog circuits for boolean satisfiability. arXiv preprint arXiv:1606.07467 (2016)