What is a universal computing machine?

Jean-Charles Delvenne

Université catholique de Louvain,
Department of Mathematical Engineering,
Avenue Georges Lemaître 4, B-1348 Louvain-la-Neuve, Belgium

Abstract

A computer is classically formalised as a universal Turing machine or a similar device. However over the years a lot of research has focused on the computational properties of dynamical systems other than Turing machines, such cellular automata, artificial neural networks, mirrors systems, etc.

In this paper we propose a unifying formalism derived from a generalisation of Turing's arguments. Then we review some of universal systems proposed in the literature and show that are particular case of this formalism. Finally, we review some of the attempts to understand the relation between dynamical and computational properties of a system.

Key words: Turing universality, dynamical systems

1 Introduction

We are interested in *computing machines*, which we informally define as machines able to solve decision problems on integers (or finite objects that can be encoded as integers), such as, for instance, primality. We are especially interested in *universal* computing machines, i.e., those who have the same power as a universal Turing machine.

Note that in this article we are only interested in solving decision problems on *integers*, while computable analysis deals with computable functions and decision problems on the reals (e.g., checking invertibility of a real-valued matrix). See for instance [29,3,25,22] on computable analysis.

Email address: jean-charles.delvenne@uclouvain.be (Jean-Charles Delvenne).

Also, we do not consider hypercomputation and systems with super-Turing capabilities, as can be found for instance in [27,4,9].

Note that here we do not define universality as the ability to 'simulate any other system'. See for instance [24] for such a notion of universality in the case of cellular automata.

We start by a quick review of computing machines, such as Turing machines, counter machines, cellular automata. We then observe, with Davis's definition of universality, that a computing machine is always a dynamical system together with an r.e.-complete halting problem. We then ask which problems can be considered as reasonable halting problems for a given dynamical system.

We then generalise Turing's famous argument to the case where a dynamical system, instead of pencil and paper, is available to a human operator. This leads to a general definition of universal computing machine. Then several definitions of universal systems are reviewed and found to be particular cases of this framework.

We finally review some results about the interaction between the computational and dynamical properties of a computing machine.

2 Turing machines

In the beginning of the twentieth century, the question arose, as a consequence of the search for foundations for mathematics, of a mechanical procedure to solve a mathematical problem, or *algorithm*. Several answers, later proved mathematically equivalent, were provided in the thirties and forties, by Post, Church, Kleene, Turing. For the original papers, see [5].

Among those answers, Turing's is perhaps the most thoroughly argued as a model for computation. Here we sketch Turing's argument to construct his machine, as it will be a basis to our definition of computing machine.

In this article, we only consider decision problems, which was not the point of view originally taken by Turing. This is not a loss of generality as computing an integer can reduced to a sequence of decision problems.

The algorithm is performed by a human operator, who applies a series of instructions on the initial data. Intermediate results are written on paper.

Turing essentially argues that the mind of the operator can only be in finitely many states, can distinguish only finitely many symbols on paper, and can write only finitely many different symbols. Hence the operator is essentially a

finite-state automaton (as far as the execution of the algorithm is concerned). The sheet of paper can be similarly considered as an unlimited linear tape divided into cells. Every cell contains a symbol out of a finite alphabet. The operator can read or write the symbol in a cell, and translate the tape one cell to the left or right. The initial data is written on finitely many cells on the tape, while the rest of the tape filled with the blank symbol. The computation ends when the operator enters the state of mind 'The computation is finished', which we call the *halting state*. As a result, a human operator executing an algorithm is modeled by a one-tape Turing machine as we know them. For a decision problem, we can also assume that the halting state of mind contains the answer to the problem, e.g., 'The computation is finished and the answer is Yes.' In this case there are two halting states. Note that a computation on a Turing machine can result in three outcomes: 'Yes', 'No' or no answer at all (i.e., the machine does not halt).

Turing confirms that his definition is sensible by showing that slightly different models, such as two-tape Turing machines, have the same power as Turing machines. That means that for every two-tape Turing machine one can construct in an effective way a one-tape Turing machine the solves the same problem, and conversely. Here by 'effective' we mean 'intuitively computable'.

He then finds that there exists a universal Turing machine, i.e., a Turing machine such that every pair (Turing machine, initial data) can be converted in an effective way into an initial data for the universal machine so as to preserve the outcome 'Yes', 'No' or 'Does not halt'.

He then proceeds to show that the *halting problem* is undecidable, more precisely r.e.-complete, as will be said later. The halting problem is the following: Given an initial data for a fixed universal Turing machine, does the universal Turing machine reach a halting state? Equivalently: Given a Turing machine and an initial data for this Turing machine, does the Turing machine reach a halting state?

3 Other universal machines

Other kinds of machines were subsequently devised to formalise computation, such as, for instance, counter machines (or register machines, or Minsky machines); see, e.g., [19]. A k-counter machine is made of k cells, each of which contains a natural integer. At every step, a finite automaton can test if the content of a counter is zero, increment a counter or decrement a counter. Again, the initial data is encoded in the content of the counters, and the computation is considered as finished when we reach a 'halting state'.

It has been proved that there exists a *universal* counter machine U.

'Universal' can be defined in terms of a reduction from the problem solved by a universal Turing machine. This means there is an effective way to encode any initial data for a fixed universal Turing machine into an initial data for the universal counter machine so as to preserve the outcome of the computation ('Yes', 'No', 'Does not halt'). Note once again that we avoid here talking about universality as 'dynamical simulation' of other dynamical systems, which avoids the need to introduce definitions of simulation.

Hence the halting problem for a universal counter machine (i.e., determining if a given initial content of the counters and an initial state of the head will reach a halting state of the head) is r.e.-complete as well.

Other similar machines were defined: Post machines, tag machines (also invented by Post) for instance; see [18]. For those again, a way to use it for computation is defined, and a universal machine is found.

All the machines above are machines with countably many states: the state of a Turing machine, for instance, is a finite sequence of symbols plus the state of the head for a Turing machine. All those machines are dynamical systems. For the moment, we loosely define a dynamical system as an object evolving in time, and completely characterised at any time by its state.

But most dynamical systems studied in mathematics and physics have an uncountable state space, e.g., cellular automata, differential equations, piecewise linear maps, etc. Examples of those systems have been proved universal. Their halting problem is imitated from the Turing machine in the following way. We choose a particular countable family of initial states, and countable family of final states, or final sets of states. Then the halting problem is given an initial state and a final state/set of states, whether the trajectory starting from the initial state will reach the final state/set of states. More specific examples are given in Section 7.

In that case, finding the relevant halting problem becomes not obvious at all, since there are many ways to select a countable family of initial states out of uncountably many. For instance, the cellular automaton of rule 110 is universal for the eventually periodic states (i.e., periodic sequences of symbols up to finitely many) but not for periodic sequences or for finite states (i.e., where all but finitely many symbols are equal to zero).

As observed in [10], some trivial dynamical systems can be also considered universal with an artificial halting problem. For instance, take the full shift on $\{0,1,2\}^{\mathbb{N}}$ is universal with the family of initial states $1^n0^t a^\infty$, where t is the halting time of a universal Turing machine on data n. If the machine does not halt on n, then the initial state is 1^n0^∞. If the machine halts on

'Yes', then $a = 1$, if it halts on 'No', then $a = 2$. Note that those states are computable: we can compute the bit of any rank for any state. Then the halting problem whether we reach the state 1^∞ from the initial state encodes the halting problem of the Turing machine. Therefore we are bound to conclude that the full shift is a universal computing machine! But it is only so with respect to a certain cooked up halting problem. It is also clear that unreasonable choices of initial conditions (i.e., undecidable) will make even simpler systems, such as the identity, even more powerful than Turing machines and shows that choosing a relevant halting problem requires caution.

As a conclusion, to every universal computing machine, is associated a certain r.e.-complete halting problem.

4 Davis universality

Davis [6], turning things around, proposed an astute definition of universality for a counter machine, Turing machine or any similar kind of object. A machine is said to be *universal* if and only if its halting problem is r.e.-complete. This definition essentially coincides with the former, but it bypasses the the mention of a universal Turing machine and the need for an effective encoding. Instead, the coding is implicit in the r.e.-completeness of the halting problem. Indeed, an r.e.-complete problem is one to which the halting problem of a universal Turing machine, and any other r.e. problem, can be reduced.

Hence a particular dynamical system is said to be universal *with respect to a certain problem*, called the halting problem, when this problem is r.e.-complete. In other terms, a computing machine is composed of a dynamical system together with a halting problem. As seen above, the choice of the natural halting problem for a dynamical system is sometimes obvious, by imitation from known examples, and sometimes more delicate.

Davis's definition makes the quest for universal computing machines a particular case of the quest for undecidable mathematical problems. It happens in mathematics that a problem occurs, that one would like to solve, but turns out to be undecidable. For instance, whether a given polynomial in several variables with integral coefficients has an integral zero is r.e.-complete (Hilbert's tenth problem, solved by Matyasevitch in 1970 [17]); whether a finitely presented group is the trivial group is r.e.-complete as well [26]. Those problems have been raised for their mathematical interest, not in order to define a new kind of computing machine. It seems difficult indeed to interpret them as the halting problem of some dynamical system.

As a conclusion, we have to solve the double question:

- Given a dynamical system, what is a relevant halting problem for it?
- What r.e.-complete problems can be considered as the halting problem of some computing machine?

5 Turing's argument revisited

In this section, we propose a recipe to address the two questions just above. We adapt Turing's argument to get a fine understanding of the interaction between dynamics and computation, and select a relevant halting problem.

Like in Turing's original argument, a human wants to solve decision problem. However, this time she has no paper or pencil, but a physical system. She doesn't necessarily know about the initial state of the system. During the process of computation, she can observe and act on the system. Like Turing, we assume that the human operator's mind can be in finitely many different states. Hence, we assume that the human can be modelled by a finite automaton, with finitely many actions on the system and finitely many possible observations from the system. This finite automaton acts as a controller on the system, in a feedback loop. Finite automaton accepting inputs (here, observations) and producing outputs (here, actuation) are also called Mealy automata, or transducers. See Fig. 1. We use the term 'state' for both the dynamical system and the controller, which models the mind. This justified as the controller is itself a dynamical system. By connecting the controller with the dynamical system, we get a new closed-loop dynamical system.

Hence a computing machine is defined in the following way.

Metadefinition 1 *A* computing machine *is defined by a dynamical system along with a countable family of controllers. Every controller is a finite state automaton with initial states and final states.*

We define the halting problem *as the following. Given a controller, the dynamical system is initialised to an arbitrary state and the controller is initialised to one of the initial states; is there a trajectory of the closed-loop system where the controller starts from an initial state and eventually reaches a final state?*

Some remarks must be made at this point.

The name 'halting problem' does not imply that the dynamical system stops after we have reached a final (or 'halting') state of the automaton. It just stops to be interesting, since we have answered the instance of the problem we wanted to solve.

Contrarily to most examples of universal systems found in the literature, there is no explicit reference to an 'initial state'. This is because a particular initial state x can be encoded in the set of actions: 'Set the state to x'. Or it can be encoded in the set of observations, in which case the computation starts with the following instruction: 'Observe the state; if it is not x then enter an infinite loop (i.e., never reach a final state)', which ensures that only computations starting by x will be processed.

We therefore see that the need for specifying an initial state is not as fundamental as it appears.

Since the controllers are finite state automata, it means that for a given controller, finitely many observations can be made on the system, and finitely many actions can be performed on it. As there are countably many controllers, only countably many possible observations and actions are to be considered for the system.

We speak of metadefinition rather than definition, because we still have to specify what is a dynamical system, what kind of observations and actuations are allowed on them, and how to interconnect the dynamical system with the controller. The answer to all these questions depends on our 'model of the world'. For instance, if we want to model physics by deterministic discrete-time systems, we'll consider a system of this kind. If we believe physical quantities cannot be observed with infinite precision, then we cannot allow the controller to observe if the system is a state x. And so on. Let us review some possibilities.

6 Dynamical systems

A dynamical system is intuitively anything that evolves in time. Many classes of dynamical systems exist, some of which we quickly describe in this section.

The most typical class is deterministic, discrete-time systems, given by an evolution map $f : X \to X$, where X is the *state space*. A state x is transformed into $f(x)$, then $f(f(x))$, and so on.

Examples include Turing machines, cellular automata, subshifts, piecewise affine maps, piecewise polynomial maps and neural networks.

Open dynamical systems (or input/output dynamical systems) allow for actuation. For instance, x is sent to $f(x, u)$, where u is the input (or actuation).

An observation on a dynamical system is most often a map $y = g(x)$, where x is the state of the system. As we want finitely many values for y, an observation is a partition of the state space of the system into finitely many sets. In

principle, we could also consider a nondeterministic relation between x and y (for instance, to model an uncertain observation), but this seems to be unexplored in the literature of computational universality.

We may also consider a nondeterministic system; for instance the state x is sent into a ball of radius ϵ around $f(x)$. This is used to model perturbations that we know are bounded by ϵ.

Continuous-time systems are usually defined by a differential equation $\dot{x} = f(x)$ on (a part of) $\mathbb{R}^n$, or $\dot{x} = f(x, u)$, where $u(t)$ is the input function. In that case, the closed-loop system is a hybrid system: a mix of continuous and discrete dynamics.

Here we do not consider quantum universal systems; see for instance [8].

7 Reachability problems

Most definitions of universality rely on a reachability problem. The *reachability problem* for a discrete-time deterministic system $f : X \to X$ goes as follows: we are given two points x and y ('point-to-point reachability') or a point x and a set Y ('point-to-set'), and the question is whether there is a t such that $f^t(x) = y$ or $f^t(x) \in Y$.

A reachability problem is modelled according to Metadefinition 1 as follows. The controller first stets the inital condition to x; then it lets the system evolves according to f; when the state of the system is y or belongs to Y, the controller jumps to its final state.

Of course, the halting problem for a universal Turing machine, counter machines and many others is a (point-to-set) reachability problem.

In cellular automata, point-to-point reachability with almost periodic configurations (made of a same pattern indefinitely repeated except for finitely many cells) is usually considered. For instance the automaton 110 and the Game of Life are universal according to this definition. Why almost periodic configurations and not a wider, or smaller, countable family of points? This is discussed in [28].

For systems in $\mathbb{R}^n$, points with rational coordinates and sets defined by polynomial inequalities with rational coefficients (e.g., polyhedra or euclidian balls) are usually considered. The choice of rational numbers seems to be guided by simplicity only.

Let us give some examples of universal systems according to this definition.

- A piecewise-affine continuous map in dimension 2 [13]. This map is defined by a finite set of affine maps of rational coefficients, on domains delimited by lines with rational coefficients.
- Artificial neural networks for several kinds of saturation functions [27].
- A closed-form analytic map in dimension 1 [14].

We can define in a very similar way universal systems in continuous time. Examples of such systems are:

- A piecewise-constant derivative system in dimension 3 [2]. The state space is partitioned on finitely domains delimited by hyperplanes with rational coefficients, and the vector field is constant with a rational value on every domain.
- A ray of light between a set of mirrors [21].
- Black hole computation, which is the interaction of signals in space-time [11].

Despite its popularity and apparent simplicity, we believe that the reachability problems as a basis to define universality suffer from several drawbacks.

First, it is possible to reach unpleasant conclusions, such as the full shift being universal, by choosing artificial initial conditions for the system, as already highlighted in Section 3.

Second, the possibility offered to the controller to set up or observe a state of the system with infinite precision seems unphysical. The least uncertainty on the initial condition can *a priori* completely destroy the computation; ensuring that a physical system is, e.g., in a rational state is obviously an impossible task in practice. It has been shown that many reachability problems become decidable when perturbation is added to the dynamics, thus killing universality; see for instance [1,16,12].

Lastly, it is difficult to prove find interesting necessary or sufficient conditions of universality base on the dynamical properties of the system, as emphasized below in Section 10.

In next sections, we see two other halting problems that generalise the halting problem of Turing machine, and avoid some of the pitfalls mentioned above.

8 Digital computing machines

Suppose we have an arbitrary symbolic dynamical system. A symbolic dynamical system is one whose state is a sequence of symbols from a finite alphabet. In other terms, the state space is $A^{\mathbb{N}}$ or $A^{\mathbb{Z}}$, for a finite alphabet A, or a closed

subset of it. Remember that such a set can be endowed with the product topology. The dynamical system is given by a continuous map on the state space.

Of course, the state space could also be $A^{\mathbb{Z}^d}$, for instance, which can be recoded into $A^{\mathbb{N}}$ by reading content of the cells in arbitrary order.

Symbolic dynamical systems include Turing machines, cellular automata and subshifts. No actuation is needed, the same map is applied at every step, the controller here is only an observer.

What is the most natural halting problem for such a dynamical systems? In other words, what is a universal *digital computing machine*?

A *cylinder* is a set of $A^{\mathbb{N}}$ of the form $wA^{\mathbb{N}}$, for any word $w \in A^*$, or a set of the form ${}^{\mathbb{N}}AwA^{\mathbb{N}}$ in $A^{\mathbb{Z}}$. Boolean combinations of cylinders are exactly the clopen (closed open sets) of the space.

We choose clopen sets as observation sets. This is a natural choice because it means that finitely many symbols are observed at every steps, and it means that a finite precision measurement is required. The only initial set is the full space; in other terms, nothing is known about the initial state of the dynamical system. Any deterministic finite automaton can be chosen as controller. This halting problem was proposed in [7]. We therefore say that a symbolic system is a universal digital computing machine if this halting problem is r.e.-complete.

It was shown [7] that a universal Turing machine is also universal for this definition, with some mild modifications. It also has the advantage to lead to non trivial conditions on the dynamical properties of the system for universality to emerge; see Section 10.

The digital computing machines show some robustness to perturbation, because a small enough perturbation on the initial condition of a successful trajectory (i.e., leading the controller to a final state) will keep the trajectory successful.

9 Actuation of dynamical systems

Turing machines are interpreted very simply as closed loop systems. Given a finite set A of symbols (including a blank symbol), we consider the set $A^{\mathbb{Z}}$ (or its restriction to finite configurations if we want a countable state space; finite configurations are those that are entirely blank except for finitely many symbols). On this set we have the following possible actuations: shift to the left, shift to the right or change the symbol in position zero to another

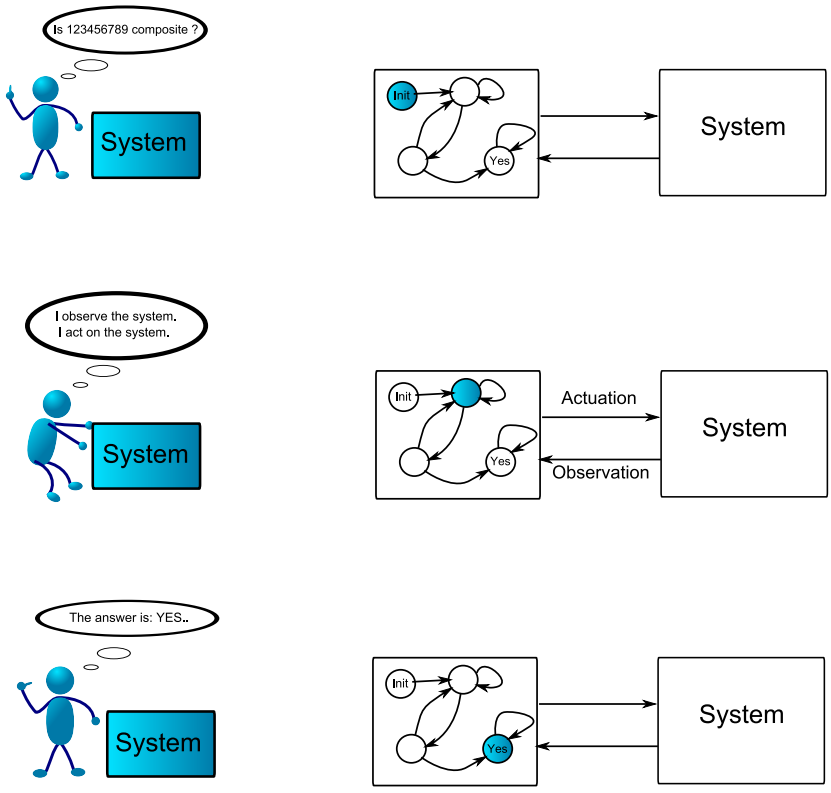

Fig. 1. Turing's argument revisited: How to compute with a dynamical system.

symbol. The only possible observation is the symbol in position zero. This is a simple open (input/output) dynamical system. If we now control it with an arbitrary finite automaton, what we get is exactly a Turing machine. This open dynamical system is therefore universal.

Despite this fundamental example, most set-ups proposed in the literature make no use of the actuation we let the dynamical system evolve by itself, without influence, except possibly at the very step to set up the initial condition. A slightly more elaborate system is proposed in [23], in which we can act on the system with three maps: f_0, f_1, f. The dynamical system starts at the origin (or any point fixed once for all), then we apply a sequence of f_0 and f_1 to introduce a binary word encoding the data; for instance, the number 100111 is encoded by the state $f_1 f_1 f_1 f_0 f_0 f_1(0)$. Then we apply f repeatedly.

10 Dynamical properties of universal systems

What is the link between the dynamics of a system and its computational capabilities?

Wolfram proposed a loose classification of 1-D cellular automata based on the patterns present in the space-time diagram of the automaton; see [30]. He then conjectured that the universal automata are in the so-called 'fourth class', associated to the most complex patterns.

Langton [15] advocated the idea of the 'edge of chaos', according to which a universal cellular automaton is likely to be neither globally stable (all points converging to one single configuration) nor chaotic. See also [20] for a discussion. Other authors argue that a universal system may be chaotic; see [27].

However it seems difficult to prove any non-trivial result of this kind with the point-to-point or point-to-set reachability definition of universality. Moreover a countable set of points can be 'hidden' in a very small part of the state space (nowhere dense, with zero measure for instance), so the link between this set and the global dynamics is unclear in general.

Digital computing machines are analysed in [7], where it is shown that a universal system according to this definition has at least one proper closed subsystem, must have a sensitive point and can be Devaney-chaotic.

11 Conclusions

A framework has been proposed to unify many of the definitions of computing machines found in the literature. A universal computing machine is defined as a dynamical system together with a suitable r.e.-complete problem; this definition is flexible with respect to the kind of dynamical systems we consider.

In particular it appears that reachability problems, despite their mathematical interest, are not the only generalisation of the Turing machine's halting problem, and perhaps not the most natural.

In particular, it appears to us that universality for open dynamical systems is an almost blank field waiting to be explored.

12 Acknowledgements

The author is indebted to José Félix Costa for pointing some references.

This paper presents research results of the Belgian Programme on Interuniversity Attraction Poles, initiated by the Belgian Federal Science Policy Office. It has been also supported by the ARC (Concerted Research Action) "Large Graphs and Networks", of the French Community of Belgium. The scientific responsibility rests with its authors. The author is holding a FNRS fellowship (Belgian Fund for Scientific Research).

References

[1] E. Asarin and A. Bouajjani. Perturbed turing machines and hybrid systems. In *Proceedings of the 16th Annual IEEE Symposium on Logic in Computer Science (LICS-01)*, pages 269–278, Los Alamitos, CA, June 16–19 2001. IEEE Computer Society.

[2] E. Asarin, O. Maler, and A. Pnueli. Reachability analysis of dynamical systems having piecewise-constant derivatives. *Theoretical Computer Science*, 138(1):35–65, 1995.

[3] L. Blum, M. Shub, and S. Smale. On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. *Bulletin of the American Mathematical Society*, 21:1–46, 1989.

[4] O. Bournez and M. Cosnard. On the computational power of dynamical systems and hybrid systems. *Theoretical Computer Science*, 168:417–459, 1996.

[5] M. Davis, editor. *The Undecidable, Basic Papers on Undecidable Propositions, Unsolvable Problems And Computable Functions.* Raven Press, 1965.

[6] M. D. Davis. A note on universal Turing machines. In C.E. Shannon and J. McCarthy, editors, *Automata Studies*, pages 167–175. Princeton University Press, 1956.

[7] J.-Ch. Delvenne, P. Kůrka, and V. D. Blondel. Computational universality in symbolic dynamical systems. *Fundamenta Informaticae*, 71:1–28, 2006.

[8] D. Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London Ser. A*, A400:97–117, 1985.

[9] Francisco Dória and José Félix Costa, editors. *Special Issue of Applied Mathematics and Computation on Hypercomputation*, volume 178. Elsevier, 2006.

[10] B. Durand and Z. Róka. The Game of Life: universality revisited. In M. Delorme and J. Mazoyer, editors, *Cellular Automata: a Parallel Model*, volume 460 of *Mathematics and its Applications*, pages 51–74. Kluwer Academic Publishers, 1999.

[11] Jérôme Durand-Lose. Abstract geometrical computation 1: embedding black hole computations with rational numbers. *Fund. Inf.*, 74(4):491–510, 2006.

[12] Peter Gács. Reliable cellular automata with self-organization. In *38th Annual Symposium on Foundations of Computer Science*, pages 90–99, Miami Beach, Florida, 20–22 October 1997. IEEE.

[13] P. Koiran, M. Cosnard, and M. Garzon. Computability with low-dimensional dynamical systems. *Theoretical Computer Science*, 132(1-2):113–128, 1994.

[14] P. Koiran and C. Moore. Closed-form analytic maps in one and two dimensions can simulate universal Turing machines. *Theoretical Computer Science*, 210(1):217–223, 1999.

[15] C. G. Langton. Computation at the edge of chaos. *Physica D*, 42:12–37, 1990.

[16] W. Maass and P. Orponen. On the effect of analog noise in discrete-time analog computations. *Neural Computation*, 10(5):1071–1095, 1998.

[17] Y. V. Matiyasevich. *Hilbert's Tenth Problem*. MIT Press, 1993.

[18] M. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, 1967.

[19] M. L. Minsky. Recursive unsolvability of Post's problem of "tag" and other topics in theory of Turing machines. *Annals of Mathematics (2)*, 74:437–455, 1961.

[20] M. Mitchell, P. T. Hraber, and J. P. Crutchfield. Dynamic computation, and the "edge of chaos": A re-examination. In G. Cowan, D. Pines, and D. Melzner, editors, *Complexity: Metaphors, Models, and Reality*, Santa Fe Institute Proceedings, Volume 19, pages 497–513. Addison-Wesley, 1994. Santa Fe Institute Working Paper 93-06-040.

[21] C. Moore. Unpredictability and undecidability in dynamical systems. *Physical Review Letters*, 64(20):2354–2357, 1990.

[22] C. Moore. Recursion theory on the reals and continuous-time computation. *Theoretical Computer Science*, 162(1):23–44, 1996.

[23] C. Moore. Dynamical recognizers: real-time language recognition by analog computers. *Theoretical Computer Science*, 201:99–136, 1998.

[24] N. Ollinger. The intrinsic universality problem of one-dimensional cellular automata. In H. Alt and M. Habib, editors, *Symposium on Theoretical Aspects of Computer Science (Berlin, Germany, 2003)*, volume 2607 of *Lecture Notes in Computer Science*, pages 632–641. Springer, Berlin, 2003.

[25] M. Boykan Pour-El and I. Richards. Computability and noncomputability in classical analysis. *Transactions of the American Mathematical Society*, 275:539–560, 1983.

[26] M. O. Rabin. Recursive unsolvability of group theoretic problems. *Annals of Math.*, 67:172–194, 1958.

[27] H. T. Siegelmann. *Neural Networks and Analog Computation: Beyond the Turing Limit*. Progress in Theoretical Computer Science. Springer-Verlag, 1999.

[28] K. Sutner. Almost periodic configurations on linear cellular automata. *Fundamenta Informaticae*, 58(3–4):223–240, 2003.

[29] K. Weihrauch. *Computable Analysis*. Springer-Verlag, 2000.

[30] S. Wolfram. *A new kind of science*. Wolfram Media, Inc., Champaign, IL, 2002.