The Limits of Decidable States on Open-Ended Evolution and Emergence*

Santiago Hernández-Orozco[1], Francisco Hernández-Quiroz[2] and Hector Zenil[3,4]

[1]Posgrado en Ciencias e Ingeniería de la Computación, UNAM, Mexico.
[2]Departamento de Matemáticas, Facultad de Ciencias, UNAM, Mexico
[3]Department of Computer Science, University of Oxford, UK
[4]Algorithmic Nature Group, LABORES, Paris, France
[1]hosant@ciencias.unam.mx, [2]fhq@ciencias.unam.mx, [3]hector.zenil@algorithmicnaturelab.org

Abstract

Using algorithmic complexity theory methods, we propose a robust computational definitions for open-ended evolution (OEE) and adaptability of computable dynamical systems. With this framework, we show that decidability imposes absolute limits to the growth of complexity on computable dynamical systems up to a logarithm of a logarithmic term. Conversely, systems that exhibit open-ended evolution must be undecidable and have irreducible behaviour through the evolution of the system. Complexity is assessed in terms of three measures: sophistication, coarse sophistication and busy beaver logical depth.

Introduction and Preliminaries

Broadly speaking, a dynamical system is one that changes over time. Prediction of the future behaviour of a dynamical system is a main issue for science generally: scientific theories are tested upon the accuracy of their predictions; and establishing invariable properties through the evolution of a system is an important goal. Limits to this predictability are known in science. For instance, chaos theory establishes the existence of systems in which small deficits in the information of the initial states makes accurate predictions of future states unattainable. However, on this document we focus on systems for which we have unambiguous, finite (on size and time) and complete descriptions of their initial states and their behaviour: computable dynamical systems.

Since their formalization by Church and Turing, the class of computable systems have shown that, even without information deficits (i.e., with complete descriptions), there are future states that cannot be predicted, in particular the state known as *halting state* (Turing, 1936). We will use this result to show how prediction imposes limits to the growth of complexity during the evolution of a system.

The relationship between dynamical systems, computability and Turing machines, along with the implied unpredictability of their behaviour, was observed by Moore (Moore, 1991) and Wolfram (Wolfram, 2002). Delvenne, Kurka and Blonde (Delvenne et al., 2006) have explored robust definitions for computable (effective) dynamical systems and universality generalizing Turing's halting states, along with conditions and implications for universality, decidability and their relationship with chaos. Undecidability of certain properties for certain analytic dynamical systems have been studied by Bournez (Bournez et al., 2013). The definitions and general approach used in this paper differ from those sources, but are ultimately related.

*The adapted definitions and complete proofs used in this article can be found at the extended version of this article `http://complexitycalculator.com/algorithmicOEE.zip`

Computable Functions

In a broad sense, an object x is *computable* if it can be described by a Turing machine (Turing, 1936); for example if there exists a Turing machine that produces x as an output. Is clear that any finite string on a finite alphabet is a computable object. Following Turing's tradition, we provide below a more formal definition.

As usual, we can define a 1 to 1 mapping between the set of all finite binary strings $\mathbb{B}^* = \{0,1\}^*$ and the natural numbers by the relation induced by the lexicographic order of the form: $\{(\text{“”}, 0), (\text{“0”}, 1), (\text{“1”}, 2), (\text{“00”}, 3), ...\}$. Using this relation we can see all natural numbers (or positive integers) as binary strings and vice versa. Accordingly all natural numbers are computable.

A string p is a *valid program* for the Turing machine T if during the execution of T with p as input all the characters in p are read. We call $T(p)$ the output of the machine, if it stops. A Turing Machine is *prefix-free* if no valid program can be a proper substring of another valid program (but can be a postfix of one). We call a valid program a *self delimited object*. Note that, given the relationship between natural numbers and binary strings, the set of all valid programs is an infinite proper subset of the natural numbers.

Formally, a function $f : \mathbb{N} \to \mathbb{N}$ is *computable* if there exist a Turing Machine T such that $f(x) = T(x)$. A Turing Machine U is called *universal* if there exist a computable function g such that for every Turing machine T there exist a string $\langle T\rangle \in \mathbb{B}^*$ such that $f(x) = U(\langle T\rangle g(x))$, where $\langle T\rangle g(x)$ is the concatenation of the strings $\langle T\rangle$ and $g(x)$.

Given the previous case, $\langle T \rangle$ and $g(x)$ are called a *codification or representations* of the function f and the natural number x, respectively. From now on we will denote by $\langle f \rangle$ and $\langle x \rangle$ the codification of f and x. The codification $g(x)$ is *unambiguous* if is injective.

For functions with more than one variable, if x is a pair $x = (x_1, x_2)$, we say that the codification $g(x)$ is unambiguous if its injective and the inverse functions $g_1^{-1} : g(x) \mapsto x_1$ and $g_2^{-1} : g(x) \mapsto x_2$ are computable. If x is a tuple $(x_1, ..., x_i, ..., x_n)$, then the codification $g(x)$ is unambiguous if the function $(x, i) \mapsto x_i$ is computable.

A sequence of strings $\delta_1, \delta_2, ..., \delta_i, ...$ is computable if the function $\delta : i \mapsto \delta_i$ is computable. A real number is computable if its decimal expansion is a computable sequence. For complex numbers and higher dimensional spaces, we say that they are computable if each of its coordinates are also computable.

Finally, for each of the described objects, we call the representation of the associated Turing machine *the representation of the object for the reference Turing machine* U, and we define computability of further objects by considering their representations. For example, a function $f : \mathbb{R} \to \mathbb{R}$ is computable if the mapping $\langle x_i \rangle \mapsto \langle f(x_i) \rangle$ is computable and we will denote by $\langle f \rangle$ the representation of the associated Turing machine, calling it the codification of f itself.

Algorithmic Descriptive Complexity

Given a prefix-free universal Turing Machine U with alphabet Σ, the *algorithmic descriptive complexity* (also known as Kolmogorov complexity and Kolmogorov-Chaitin complexity (Kolmogorov, 1965; Chaitin, 1982) of a string $s \in \Sigma^*$ is defined as

$$K_U(s) = \min\{|p| : U(p) = s\},$$

where U is a universal prefix-free Turing Machine and $|p|$ is the number of characters of p.

The algorithmic descriptive complexity measures the minimum amount of information needed to fully describe a computable object within the framework of a universal Turing machine U. If $U(p) = s$ then the program p is called a description of s, the first of the smallest descriptions (on alphabetical order) is denoted by s^* and by $\langle s \rangle$ a non necessarily minimal description computable over the class of objects. If M is a Turing machine, a program p is a description or codification of M for U if for every string s we have that $M(s) = U(p\langle s \rangle)$. In the case of numbers, functions, sequences and other computable objects we consider the descriptive complexity of its smallest description. For example, for a computable function $f : \mathbb{R} \to \mathbb{R}$, $K(f)$ is defined as $K(f^*)$ where $f^* \in \mathbb{B}^*$ is the first of the minimal descriptions for f.

Of particular importance for this document is the *conditional descriptive complexity*, which is defined as:

$$K_U(s|r) = \min\{|p| : U(pr) = s\},$$

where pr is the concatenation of p and r. This measure can be interpreted as the *smallest amount of information needed to describe s given a full description of r*. We can think of p as a program with input r.

One of the most important properties of the descriptive complexity measure is its *stability*: the difference between the descriptive complexity of an object, given two universal Turing machines, is at most constant. Therefore the reference machine U is usually omitted in favor of the *universal measure* K. From now on we will omit the subscript from the measure.

Randomness A string x is known as r*-random* or *incompressible* if $K(x) \geq |x| - r$. This definition states that a string is random if it does not has a significantly shorter complete description than the string itself. A simple counting argument shows the existence of random strings. Now, is easy to verify that every string x has a self delimited computable unambiguous codification with strings of the form $1^{\log|s|}0|s|s$ (Li and Vitányi, 1997, section 1.4). Therefore, there exist a natural r such that if x is r-random then $K(x) = |x| - r + O(\log|x|)$, where $O(\log|x|)$ is a positive term. We will say that such strings hold the randomness inequality *tightly*.

Let M be a halting Turing Machine with description $\langle M \rangle$ for the reference machine U. A simple argument can show that the halting time of M cannot be a large *random* number: let U^H be a Turing Machine that emulates U while counting the number of steps, returning the execution time upon halting; if r is a large random number then M cannot stop in time r, otherwise the program $\langle U^H \rangle \langle M \rangle$ will give us a *short* description of r. This argument is summarized by the following inequality:

$$K(T(M)) \leq K(M) + O(1), \tag{1}$$

where $T(M)$ is the number of steps that took the machine M to reach the halting state, the *execution time* of the machine M.

Computable Dynamical Systems

Formally, a *dynamical system* is a rule of evolution in time within a state space; space that is defined as the set of all possible states of the system (Meiss, 2007). For this work we will focus in a functional model for dynamical systems with a constant initial state and variables representing the previous state and the time of the system. This model allows us to set halting states for each time on a discrete scale in order to study the impact of the descriptive complexity of time during the evolution of a discrete computable system.

A deterministic discrete space system is defined by an *evolution function (or rule)* of the form $M_{t+1} = S(M_0, t)$, where M_0 is called the *initial state* and t is a positive integer called the *time* variable of the system. The sequence of

states $M_0, M_1, ..., M_t, ...$ is called the evolution of the system. Given a reference universal Turing Machine U, if S is a computable function and M_0 is a computable object, we will say that S is a *computable dynamical system*. An important property of computable dynamical systems is the uniqueness of the successor state which implies that equal states must evolve equally given the same evolution function, in other words:

$$M_t = M_{t'} \implies M_{t'+1} = M_{t+1}. \quad (2)$$

The converse is not necessarily true.

Now, a *complete description of a computable system* $S(M_0, t)$ should contain enough information to compute the state of the system at any time and hence it must entail the codification of its evolution function S and a description of the initial state M_0, which is denoted by $\langle M_0 \rangle$. As a consequence, if we only describe the system at time t by a codification of M_t, then we do not have enough information to compute the successive states of the system. So we will define the *complete description* of a computable system at the time t as a unambiguous codification of the ordered pair composed by $\langle S \rangle$ and $\langle M_t \rangle$, i.e. $\langle (S, \langle M_t \rangle) \rangle$, with $\langle (S, \langle M_0 \rangle) \rangle$ representing the initial state of the system. It is important to note that, for any computable and unambiguous codification function g of the stated pair, we have

$$K(\langle (S, \langle M_t \rangle) \rangle) \leq K(S) + K(M_0) + K(t) + O(1);$$

as we can write a program that uses the descriptions for S, M_0 and t to find the parameters and then evaluate $S(M_0, t)$, finally producing M_t.

Open-Ended Evolution in Computable Dynamical Systems

Defining and establishing the properties required for a system to exhibit **Open-ended evolution (OEE)** is considered an open question (Bedau et al., 2000; Soros and Stanley, 2014; Standish, 2003) and OEE has been proposed as a required property of evolutionary systems capable of producing life (Ruiz-Mirazo et al., 2002). This has been implicitly verified by various experiments *in-silico* (Lindgren, 1992; Adami and Brown, 1994; Lehman and Stanley, 2008; Auerbach and Bongard, 2014).

A line of thought posits that open-ended evolutionary systems tend to produce families of objects of increasing *complexity* (Bedau, 1998; Auerbach and Bongard, 2014). Following this idea, OEE in a computable dynamical system can be characterized as a process that has the property of producing families of objects of increasing *complexity*. Formally, given a *complexity measure* C, we say that a computable dynamical system S exhibits *open-ended evolution* with respect to C if for every time t there exists a time t' such that the complexity of the system at the time t' is greater than the complexity at the time t, i.e. $C(S(M_0, t)) < C(S(M_0, t'))$, where a complexity measure is a (not necessarily computable) function that goes from the state space to a positive numeric space.

The existence of such systems is trivial for complexity measures on which any infinite set of the natural numbers (not necessarily computable) contains a subset where the measure grows strictly:

Lemma 1. *Let C be a complexity measure such that any infinite set of natural numbers has a subset where C grows strictly. Then a computable system $S(M_0, t)$ is a system that produces an infinite number of different states if and only if it exhibits OEE for C.*

Given the previous lemma, a trivial computable system that simply produces all the strings in order exhibits OEE on a class of complexity measures that includes algorithmic description complexity. However, intuitively, we conjecture that such systems have a much simple behaviour compared to what we observe on the natural world and the cited artificial life systems. We can avoid some of these issues with a stronger version of OEE.

Definition 2. A sequence of naturals $n_0, n_1, ..., n_i, ...$ exhibits *strong open-ended evolution* (strong OEE) with respect to a complexity measure C if for every index i there exists an index i' such that $C(n_i) < C(n_{i'})$, and the complexity of the sequence $C(n_0), C(n_1), ..., C(n_i), ...$ does not drop significantly, i.e. $i \leq j$ implies $C(n_i) \leq C(n_j) + \gamma(j)$ where $\gamma(j)$ is a positive function that does not grow *significantly*.

It is important to note that, while the definition of OEE allows significant drops of complexity during the evolution of a system, strong OEE requires for the complexity of the system to not decrease *significantly* during its evolution. In particular we will ask for $C(n_j) - \gamma(j)$ to not be upper-bounded for any infinite subsequence.

Various complexity measures have been proposed that assign low complexity to an infinity of natural numbers deemed to be *simple*. Two examples of such measures are logical depth (Bennett, 1988) and sophistication (Koppel, 1988). Nonetheless, if C is a complexity measure capable of measuring OEE then there must exist infinite sets where C grows strictly. A trivial counting argument shows that the algorithmic descriptive complexity is unbounded in any infinite set, therefore is also unbounded in any set where any other complexity measure grows strictly. Formally:

Lemma 3. *If a system S exhibits OEE (and strong OEE) for a complexity measure C then it also shows OEE with respect to the descriptive complexity K.*

Given the previous lemma, the results shown in the next section can also be extended to any other complexity measure capable of showing OEE.

A Computational Model for Adaptation

Lets start by characterizing the evolution of an organism or a population by a computable dynamical system. It has been argued that, in order for *adaptation* and survival to be possible, an organism must contain a representation of the environment so that, given a reading of the representation, the organism can choose a behaviour accordingly (Zenil et al., 2012). The more approximate this representation is, the better the organism is adapted to its environment. If the organism is computable, this information can be codified by a computable structure. We will denote this structure by M_t, where t stands for the time corresponding to each of the stages of the evolution of the organism. This information is then processed following a finitely specified unambiguous set of rules that, in finite time, will determine the adapted behaviour of the organism according to the information codified by M_t. We will denote this behaviour (or a theory explaining it) with the program p_t. An adapted system is one that produces an acceptable approximation of its environment. An environment can also be represented by a computable structure E. In other words, the system is adapted if $p_t(M_t)$ produces E. Based on this idea we propose a robust, formal definition for adaptation:

Definition 4. Let K be the prefix-free descriptive complexity. We say that the system at the state M_n is *ϵ-adapted* to the E if:

$$K(E|S(M_0, E, n)) \leq \epsilon. \tag{3}$$

The inequality states that the minimal amount of information that is needed to describe E from a complete description of M_n is ϵ or less. This information is provided in form of a program p that produces E from the system at the time n. We will define such programs p as the *adapted behaviour* of the system. Uniqueness for p is not required.

The proposed structure for adapted systems is robust since $K(E|S(M_0, E, n))$ is equal or less than the numbers of characters needed to describe any computable method of describing E from the state of the system at the time n, either be a computable theory for adaptation or a computable model for an organism that tries to predict E. Follows that any computable characterization of adaptation that can be described within ϵ number of bits meets the definition of *ϵ-adapted* given suitable choice of E, the *adaptation condition* for any given environment.

As a simple example, we can think of an organism that must find the food located at the coordinates (x, j) on a grid in order to survive. If the information of an organism is codified by a computable structure M (such as DNA), and there is a set of finitely specified, unambiguous rules that govern how this information is used (such as the ones specified by biochemistry and biological theories) codified by a program p, then we say that the organism finds the food if $p(M) = (j, k)$. If $|\langle p \rangle| \leq \epsilon$, then the we say that the organism is adapted according to a behaviour that can be described within ϵ characters. The proposed model for adaptation is not limited to such simple interactions. For a start, we can suppose that the organism *sees* a grid, denoted by g, of size $n \times m$ with food at the coordinates (j, k). The environment can be codified as a function E such that $E(g) = (j, k)$ and ϵ-adapted implies that the organism defined by the genetic code M, which is interpreted by a theory or behaviour written on ϵ bits, is capable of finding the food upon seeing g. Similarly, more complex computational structures and interactions imply ϵ-adaptation.

Now, describing an evolutionary system that (eventually) produces an *ϵ-adapted* system is trivial via an enumeration machine (the program that produces all the natural numbers in order), as it will eventually produce E itself; moreover, we want for the output of our process to remain adapted. Therefore we propose an stronger condition called *convergence*:

Definition 5. Given the description of a computable dynamical system $S(M_0, E, t)$ where $t \in \mathbb{N}$ is the variable of time, M_0 is an initial state and E is an environment, we say that the system S *converges* towards E with degree ϵ if there exist δ such that $t \geq \delta$ implies $K(E|S(M_0, E, t)) \leq \epsilon$.

For a fixed initial state M_0 and environment E, is easy to see that the descriptive complexity of a state of the system depends mostly on t: we can describe a program that, given full descriptions of S, E, M_0 and t finds $S(M_0, E, t)$, therefore

$$\begin{aligned} K(S(M_0, E, t)) \leq K(S) + K(E) + \\ K(M_0) + K(t) + O(1), \end{aligned} \tag{4}$$

where the constant term is the length of the program described. In order words, as the time t grows, *time is the main driver of the descriptive complexity within the system.*

Non-Randomness of Decidable Convergence Times

One of the most important issues for science is the prediction of the future behaviour of dynamical systems. The prediction that we focus on is that of the first state of convergence (definition 5): Will a system converge and how long it will take? In this section we show what the first limit that decidability imposes to the complexity of the first convergent state is. A consequences of this is the existence of undecidable adapted states.

Formally, for the convergence of a system S with degree ϵ to be decidable there must exist an algorithm D_ϵ such that $D_\epsilon(S, M_0, E, \delta) = 1$ if the system is convergent at the time δ and 0 otherwise. Moreover, we can describe a machine P such that given full descriptions of D_ϵ, S and M_0 it runs D_ϵ with inputs S and M_0 while running over all the possible times t, returning the first t for which the system converges. Note that $\delta = P(\langle D_\epsilon \rangle \langle S \rangle \langle M_0 \rangle \langle E \rangle)$, hence we have a short description of δ and therefore δ *cannot be random*: if $S(M_0, E, t)$ is a convergent system then

$$K(\delta) \leq K(D_\epsilon) + K(S) + K(E) + K(M_0) + O(1), \tag{5}$$

where δ is the first time at which convergence is reached. Note that all the variables are known at the initial state of the system. This result can be resumed by the following lemma:

Lemma 6. *Let S be a system convergent at the time δ. If δ is considerably more descriptively complex than the system and the environment, i.e. for every reasonably large natural number d we have that*

$$K(\delta) > K(S) + K(E) + K(M_0) + d,$$

then δ cannot be found by an algorithm described within d number of characters.

We call such times *random convergence times* and the state of the system M_δ a *random state*. It is important to notice that the descriptive complexity of a random state must be also high:

Lemma 7. *Let S be a convergent system with a complex state $S(M_0, E, \delta)$. For every reasonably large d we have that*

$$K(S(M_0, E, \delta)) > K(S) + K(E) + K(M_0) + d.$$

In other words, if δ has high descriptive complexity, then there does not exist a reasonable algorithm that finds it even if we have a complete description of the system and its environment. It follows that the descriptive complexity of a computable convergent state cannot be much greater than the descriptive complexity of the system itself.

What a *reasonably large d* is has been handled so far with ambiguity as it represents the descriptive complexity of any computable method D_ϵ. We might intend to find convergent times, which intuitively cannot be arbitrarily large. It is easy to '*cheat*' on the inequality 5 by including in the description of the program D_ϵ the full description of the convergence time δ, which is why we ask for *reasonable* descriptions.

Another question left to answer is whether complex convergent times do exist for a given limit d, considering that the limits imposed by the inequality 5 loosen up in direct relation to the descriptive complexity of S, E and M_0.

The next result answers both questions by proving the existence of complex convergent times for a broad characterization of the size of d:

Lemma 8 (Existence of Random Convergence Times)**.** *Let F be a total computable function. For any ϵ, there exist a system $S(M_0, E, t)$ such that the convergence times are $F(S, M_0, E)$-random.*

Let us focus on what the previous lemma is saying: F can be any computable function. It can be a polynomial or exponential function with respect to the length of a given descriptions for M_0 and E. It can also be any computable theory that we might propose for setting an upper limit to the size of an algorithm that finds convergence times given descriptions of the system behaviour, environment and initial state. In other words, for a class of dynamical systems, finding convergence times, therefore convergent states, is not decidable even with complete information of the system and its initial state.

Randomness of Convergence in Dynamic Environments.

So far we have limited the discussion to fixed environments. However, as observed in the physical world, the environment itself can change over time. We call such environments *dynamic environments*. In this section we extend the previous results to cover environments that change depending on time as well as on the initial state of the system. We also propose a weaker convergence condition called *weak convergence* and propose a necessary (but not sufficient) condition for the computability of convergent times called *descriptive differentiability*.

We can think of an environment E as a dynamic computable system, a moving target that also changes with time and depends on the initial state M_0. In order for the system to be convergent, we propose the same criterion: there must exist δ such that $n \geq \delta$ implies

$$K(E(M_0, n)|S(M_0, E(M_0, n), n)) \leq \epsilon. \tag{6}$$

A system with a dynamic environment also meets the inequality 5 and lemmas 6 and 8 since we can describe a machine that run both S and E for the same time t.

Now with dynamic environments, E is a moving target and therefore is convenient to consider an *adaptation period* for the new states of E:

Definition 9. We say that S *converges weakly* to E if there exist an infinity of times δ_i such that

$$K(E(M_0, \delta_i)|S(M_0, E(M_0, \delta_i), \delta_i)) \leq \epsilon. \tag{7}$$

As direct consequence of the inequality 5 and lemma 8 we have the following lemma:

Lemma 10. *Let $S(M_0, E(M_0, t), t)$ be a weakly converging system. Any decision algorithm $D_\epsilon(S, M_0, E, \delta_i)$ can only decide the first non-random time.*

As noted above, these results do not change when dynamic environments are considered. In fact, we can think of static environments as a special case of dynamic environments. However, with different targets of adaptability and convergence, it makes sense to generalize beyond the first convergence time. Also, it should be noticed that specifying a convergence index adds additional information that a decision algorithm can potentially use.

Lemma 11. *Let $S(M_0, E(M_0, t), t)$ be a weakly converging system with an infinity of random times such that $k > j$ implies that $K(\delta_k) = K(\delta_j) + \Delta K_\delta(j, k)$, where ΔK_δ*

is a (not necessarily computable) function with range on the positive integers. If the function $\Delta K_\delta(i, i+m)$ is unbounded with respect to i then any decision algorithm $D_\epsilon(S, M_0, E, i)$, where i is the i-th convergence time, can only decide a finite number of i's.

One direct consequence of the previous lemma is that if a sequence of times $\delta_1, \delta_2, ..., \delta_i, ...$ is decidable then for every m there exist a constant $c_{\delta,m}$ such that

$$\Delta K_\delta(i, i+1) = K(\delta_{i+m}) - K(\delta_i) \leq c_{\delta,m}.$$

which we can be generalized as:

Definition 12. Let $\delta_1, \delta_2, ..., \delta_i, ...$ be a strictly growing sequence of natural numbers. We define the *descriptive derivative* of the natural mapping $\delta : i \mapsto \delta_i$ as

$$\Delta K_\delta(m) = \min\{c : |K(\delta_{i+m}) - K(\delta_i)| \leq c, i \in \mathbb{N}\}. \quad (8)$$

As a direct consequence of lemma 11, the existence of a descriptive derivative is a necessary condition for the computability of δ; thus not meeting this property is sufficient, but not necessary, for undecidability. Therefore the existence of a descriptive derivative is a stronger condition which we will call *non-descriptively differentiable*.

Definition 13. We say that a sequence of times is $\delta_1, \delta_2, ..., \delta_i, ...$ is *non-descriptively differentiable* if $\Delta K_\delta(m)$ is not a total function.

Irreducibility of Descriptive Time Complexity

At the previous section, it was established that time was the main factor in the descriptive complexity of the states within the evolution of a system. This result is expanded by the time complexity stability theorem (14). This theorem establishes that, within an algorithmic descriptive complexity framework, similarly complex initial states must evolve into similarly complex future states over similarly complex time frames, effectively erasing the difference between the complexity of the state of the system and the complexity of the corresponding time and establishing absolute limits to the reducibility of future states.

Let $F(t) = T(S(M_0, E, t))$ be the *real execution time* of the system at time t. By using our time counting machine U^H it is easy to see that $F(t)$ is computable and, by uniqueness of successor state, F increases strictly with t, and hence it is injective. Consequently, F has a computational inverse F^{-1} over its image. Therefore, we have that (up to a small constant) $K(F(t)) \leq K(F) + K(t)$ and $K(t) \leq K(F^{-1}) + K(F(t))$. Follows that, $K(t) = K(F(t)) + O(c)$, where c is an integer independent of t (but that can depend on S). In other words, for a fixed system S, the execution time and the system time are *equally complex up to a constant*. From here on I will not differentiate between the complexity of both times. A generalization of the previous equation is given by the following theorem:

Theorem 14 (Time Complexity Stability)**.** *Let S and S' be two computable systems and t and t' the first time where each system reaches the states M_t and $M'_{t'}$ respectively, then exist c such that $|K(M_t) - K(t)| \leq c$ and $|K(M_t) - K(M'_{t'})| \leq c$.*

Beyond Halting States: Open-Ended Evolution

Inequality 5 states that being able to predict or recognize adaptation imposes a limit to the descriptive complexity of the first adapted state. A particular case is the halting state, as shown at the proof of lemma 8. By lemma 3 this result holds for any complexity measure. In this section we extend the lemma to continuously evolving systems, showing that computability of adapted times limits the complexity of adapted states beyond the first, imposing a limit to open-ended evolution for three complexity measures: sophistication, coarse sophistication and busy beaver logical depth.

For a system in constant evolution converging to a dynamic environment, the lemma 11 imposes a limit to the growth of descriptive complexity of a system with computable adapted states: *if the growth of the descriptive complexity of a sequence of convergent times is unbounded in the sense of definition 13 then all but a finite number of times are undecidable.* The converse would be convenient, however it is not always true. Moreover, the next series of result shows that imposing such limit would impede strong OEE:

Theorem 15. *Let S be a non cyclical computable system with initial state M_0, E a dynamic environment and $\delta_1, ..., \delta_i, ...$ a sequence of times such that for each δ_i there exist a total function p_i such that $p_i(M_{\delta_i}) = E(\delta_i)$. If the function $p : i \mapsto p_i$ is computable, then the function $\delta : i \mapsto \delta_i$ is computable.*

The last result can be applied naturally to weakly convergent systems (9): the way each adapted state approaches to E is unpredictable, in other words, its *behaviour* changes over different stages unpredictably. Formally:

Corollary 16. *Let $S(M_0, E, t)$ be a weakly converging system with adapted states $M_{\delta_1}, ..., M_{\delta_i}, ...$ and $p_1, ..., p_i, ...$ their respective adapted behaviour. If the mapping $\delta : i \mapsto \delta_i$ is non-descriptively differentiable then the function $p : i \mapsto p_i$ is not computable.*

While asking for totality might look like an arbitrary limitation at first glance, the reader should recall that in weakly convergent systems the program p_i represents an organism, a theory or other computable system that uses M_{δ_i}'s information to predict the behaviour of $E(\delta_i)$, and if this prediction does not process its environment in a sensible time frame then it is hard to argue that it represents an *adapted system* or an *useful theory*.

The intuition behind classifying descriptively differentiable adapted time sequences as *less complex* is better explained by borrowing ideas developed by Bennett and Koppel within the framework of logical depth (Bennett, 1988)

and sophistication (Koppel, 1988), respectively. Their argument states that random strings are as simple as very regular strings, given that there is no complex underlying structure in their minimal descriptions. The intuition that random objects contain no useful information leads us to the same conclusion. And thanks to theorem 14, the states must retain a high degree of randomness for random times.

Logical depth works under the assumption that complex or *deep* natural numbers take a long time to compute from near minimal descriptions. Conversely, random or incompressible strings are shallow since their minimal descriptions must contain the full description *verbatim*. For this work we will use a variation of logical depth called busy beaver logical depth, denoted by $depth_{bb}(x)$.

Sophistication is a measure of *useful information* within a string proposed by Koppel. The idea behind is to divide the description of a string x in two parts: the program that represents the *underlying structure* of the object and the input, which is the random or *structureless* component of the object. This function is denoted by $soph_c(x)$, where c is a natural number representing the significance level.

Now, the images of a mapping $\delta : i \mapsto \delta_i$ already have the form $\delta(i)$, where δ and i represent the structure and the random component respectively. Random strings should hold strongly this structure up to a logarithmic error, which is proven in the next lemma.

Lemma 17. *Let $\delta_1, ..., \delta_i, ...$ be a sequence of different natural numbers and r a natural number. If the function $\delta : i \mapsto \delta_i$ is computable then there exists an infinite subsequence where the sophistication is bounded up to a logarithm of a logarithmic term of their indexes.*

Now, small changes in the significance level of sophistication can have a large impact on the sophistication of a given string. Another possible issue is that the constant proposed at lemma 17 could appear to be large at first (but it becomes comparatively smaller as i grows). A *robust* variation of sophistication called coarse sophistication (Antunes and Fortnow, 2003), incorporates the significance level as a penalty. The definition presented here differs slightly from theirs in order to maintain congruence with the chosen prefix-free universal machine and to avoid negative values. This measure is denoted by $csoph(x)$.

With a similar argument as the one used to prove lemma 17, it is easy to show that coarse sophistication is similarly bounded up to a logarithm of a logarithmic term.

Lemma 18. *Let $\delta_1, ..., \delta_i, ...$ be a sequence of different natural numbers and r a natural number. If the function $\delta : i \mapsto \delta_i$ is computable then there exist an infinite subsequence where the coarse sophistication is bounded up to a logarithm of a logarithmic term.*

Another proposed measure of complexity is Bennett's logical depth, for which the next result follows from a theorem found by Antunes and Fortnow (Antunes and Fortnow, 2003) and lemma 18.

Corollary 19. *Let $\delta_1, ..., \delta_i, ...$ be a sequence of different natural numbers and r a natural number. If the function $\delta : i \mapsto \delta_i$ is computable then there exist an infinite subsequence where the busy beaver logical depth is bounded up to a logarithm of a logarithmic term of their indexes.*

Let us focus on the consequence of lemmas 17 and 18 and corollary 19. Given the relationship established between descriptive time complexity and the corresponding state of a system (theorem 14), these last results imply that either the complexity of the adapted states of a system (using any of the three complexity measures) grows very slowly for an infinity subsequence of times (becoming increasingly common up to probability of 1 (Calude and Stay, 2006)) or the subsequence of adapted times is undecidable. Formally:

Theorem 20. *If $S(M_0, E, t)$ is a weakly converging system with adaptation times $\delta_1, ..., \delta_i, ...$ then there exist a constant c such that, if $csoph$, $depth_{bb}$ or $soph_c$ show strong OEE that grows faster than $O(\log \log i)$ then the mapping $\delta : i \mapsto \delta_i$ is not computable.*

Technically theorem 20 does not impose undecidability to strong OEE. However, the growing rate that decidability imposes is extremely slow. If we disregard this increasingly insignificant growing rate, we can say that strong open-ended evolution implies undecidability of the adapted states.

Furthermore, by theorem 16, the behaviour and interpretation of the system evolves in an unpredictable way, establishing one path for *emergence*: *a set of rules that cannot be reduced to an initial set of rules*. Recall that for a given weakly converging dynamical system, the sequence of programs p_i represents the behaviour or interpretation of each adapted state M_i. If a system exhibits strong OEE with respect to the complexity measures $soph_c$, $csoph$ or $depth_{bb}$, by corollary 16 and theorem 20 the sequence of behaviours is uncomputable and therefore, irreducible to any function of the form $p : i \mapsto p_i$, even when possessing complete descriptions for the behaviour of the system, its environment and its initial state.

Conclusions

We have presented a formal and general mathematical model for adaptation within the framework of computable dynamical systems. This model exhibits universal properties for all computable dynamical systems, of which Turing machines are a subset.

Among other results, we have given formal definitions of open-ended evolution (OEE) and strong open-ended evolution. We also showed that decidability imposes universal limits to the growth of complexity of computable systems as measured by sophistication, coarse sophistication and busy beaver logical depth. Furthermore, as a direct implication of corollary 16 and theorem 20, undecidability of adapted states and unpredictability of the behaviour of the system

at each state is a requirement for a system to exhibit strong open-ended evolution (up to a $O(\log \log t)$ term) with respect to the complexity measures known as sophistication, coarse sophistication and busy beaver logical depth, establishing a rigorous proof that undecidability and irreducibility of future behaviour is a requirement for the growth of complexity among the class of computable dynamical systems.

Acknowledgments We would like to thank Carlos Gershenson García for his comments during the development of this project and support from grants CB-2013-01/221341 and PAPIIT IN113013.

References

Adami, C. and Brown, C. T. (1994). Evolutionary learning in the 2D artificial life system avida. In *Proc. Artificial Life IV*, pages 377–? MIT Press.

Antunes, L. and Fortnow, L. (2003). Sophistication revisited. In *ICALP: Annual International Colloquium on Automata, Languages and Programming*.

Auerbach, J. E. and Bongard, J. C. (2014). Environmental influence on the evolution of morphological complexity in machines. *PLoS Computational Biology*, 10(1).

Bedau (1998). Four puzzles about life. *ARTLIFE: Artificial Life*, 4.

Bedau, McCaskill, Packard, Rasmussen, Adami, Green, Ikegami, Kaneko, and Ray (2000). Open problems in artificial life. *ARTLIFE: Artificial Life*, 6.

Bennett, C. H. (1988). Logical depth and physical complexity. In Herken, R., editor, *The Universal Turing Machine: A Half-Century Survey*, pages 227–257. Oxford University Press.

Bournez, O., Graça, D. S., Pouly, A., and Zhong, N. (2013). *The Nature of Computation. Logic, Algorithms, Applications: 9th Conference on Computability in Europe, CiE 2013, Milan, Italy, July 1-5, 2013. Proceedings*, chapter Computability and Computational Complexity of the Evolution of Nonlinear Dynamical Systems, pages 12–21. Springer Berlin Heidelberg, Berlin, Heidelberg.

Calude, C. and Jugensen, H. (2005). Is complexity a source of incompleteness? *ADVAM: Advances in Applied Mathematics*, 35.

Calude, C. S. and Stay, M. (2006). Most programs stop quickly or never halt. *CoRR*, abs/cs/0610153.

Chaitin, G. J. (1982). Algorithmic information theory. In *Encyclopaedia of Statistical Sciences*, volume 1, pages 38–41. Wiley.

Delvenne, J.-C., Kurka, P., and Blondel, V. D. (2006). Decidability and universality in symbolic dynamical systems. *Fundam. Inform*, 74(4):463–490.

Kolmogorov, A. (1965). Three approaches to the quantitative definition of information. *Problems Inform. Transmission*, 1:1–7.

Koppel, M. (1988). Structure. In Herken, R., editor, *The Universal Turing Machine: A Half-Century Survey*, pages 435–452. Oxford University Press.

Lehman, J. and Stanley, K. O. (2008). Exploiting open-endedness to solve problems through the search for novelty. In Bullock, S., Noble, J., Watson, R. A., and Bedau, M. A., editors, *ALIFE*, pages 329–336. MIT Press.

Li, M. and Vitányi, P. (1997). *An introduction to Kolmogorov complexity and its applications*. Springer, 2nd edition.

Lindgren, K. (1992). Evolutionary phenomena in simple dynamics. In Langton, C. G., Taylor, C., Farmer, J. D., and Rasmussen, S., editors, *Artificial Life II*, pages 295–312. Addison-Wesley, Redwood City, CA.

Meiss, J. (2007). Dynamical systems. *Scholarpedia*, 2(2):1629. revision #121407.

Moore, C. (1991). Generalized shifts: Unpredictability and undecidability in dynamical systems. *Nonlinearity*, 4(2):199–230.

Ruiz-Mirazo, K., Peretó, J., and Moreno, A. (2002). A universal definition of life: Autonomy and open-ended evolution. *Origins of life and evolution of the biosphere*, 34(3):323–346.

Soros, L. B. and Stanley, K. O. (2014). Identifying necessary conditions for open-ended evolution through the artificial life world of chromaria. In *Fourteenth International Conference on the Synthesis and Simulation of Living Systems (ALIFE 14)*. MIT Press.

Standish, R. K. (2003). Open-ended artificial evolution. *International Journal of Computational Intelligence and Applications*, 3(2):167–175.

Turing, A. M. (1936). On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42:230–265.

Wolfram, S. (2002). *A New Kind of Science*. Wolfram Media Inc.

Zenil, H., Gershenson, C., Marshall, J. A. R., and Rosenblueth, D. A. (2012). Life as thermodynamic evidence of algorithmic structure in natural environments. *Entropy*, 14(11).