

On the Dynamic Qualitative Behavior of Universal Computation

Hector Zenil

*Department of Computer Science/Kroto Research Institute
The University of Sheffield
Regent Court, 211 Portobello, S1 4DP, UK
h.zenil@sheffield.ac.uk*

The possible connections between the dynamic behavior of a system and Turing universality in terms of the ability of the system to (effectively) transmit and manipulate information are explored. Some arguments will be provided using a defined compression-based transition coefficient, which quantifies the sensitivity of a system to being programmed. In the same spirit, a list of conjectures concerning the ability of busy beaver Turing machines to perform universal computation will be formulated. The main working hypothesis is that universality is deeply connected to the qualitative behavior of a system, particularly to its ability to react to external stimulus—as it needs to be programmed—and to its capacity for transmitting this information.

1. Introduction

In [1], an investigation of the dynamic properties of computing machines using a general lossless compression approach led to reasonable classifications of one-dimensional cellular automata (CAs) and other systems corresponding to Wolfram's four classes of behavior [2]. In the spirit of other analytical concepts for scale predictability (e.g., Lyapunov exponents), but employing different means, this compression-based method also led to the definition of a phase transition coefficient as a way of detecting a system's (in)stability in relation to its initial conditions and of measuring its dynamic ability to carry information. A conjecture was introduced relating the magnitude of this coefficient and the capability and efficiency with which a system performs universal computation. In this paper the conjecture is developed further with some additional arguments.

In [3], a related conjecture was presented concerning other kinds of simply defined programs, establishing that all busy beaver Turing machines may be capable of universal computation, as they seem to share some of the informational and complex properties of systems capable of universal computational behavior. The conjecture will be regarded in light of algorithmic complexity, particularly of Bennett's logical depth [4], and will be reconnected to the first conjecture via

the dynamical properties of these machines through the compression-based phase transition coefficient.

Some definitions of concepts to be discussed, either as foundations of these possible new connections or as evidence for making such claims, will be introduced first. The investigation is meant to be an exploration of empirical observations through quantitative measures that attempt to capture qualitative properties of the dynamic behavior of systems capable of computational universality.

1.1 Preliminaries

Proof-of-universality results for simple programs have traditionally relied on localized structures (or “particles”) instead of relatively uniform regions. This means that a measure of entropy of a system will tend to be below its theoretical maximum. At the same time, however, this “particle-like” behavior is, and must in principle be, unpredictable for the system to reach computational universality.

S. Wolfram has classified all one-dimensional nearest neighborhood CAs into four classes [2]: class 1, ordered behavior; class 2, periodic behavior; class 3, random or chaotic behavior; and class 4, complex behavior. The first two are totally predictable. Random CAs are unpredictable. Somewhere in between, in the transition from periodic to chaotic, complex and interesting behavior can occur.

One of Wolfram’s open problems [5] in CAs, for example, is the question of the computational universality of a class 3 system (random-looking, such as rule 30) for which an entropy measure remains near its maximum at every time step, and which is unlikely to show any particle-like behavior. The question is whether such a “hot system” can carry information and be programmed. The techniques to prove that such a system is universal may require different methods from those hitherto used for systems in which structures can be distinguished and which can therefore be made to carry information through them. The common belief is that these kinds of systems may be powerful enough but are just too complicated—perhaps even impossible—to program. The encoding required to deal with the sophistication of a class 3 CA would itself probably have to possess the sophistication of a computationally universal system. This brings us to Wolfram’s Principle of Computational Equivalence (PCE), which states that almost all processes that are not obviously simple can be viewed as computations of equivalent sophistication ([2], pp. 5 and 716–717).

1.2 The Behavior of Simple Programs

In 1970, Conway invented an automaton that was popularized by Gardner [6] and was known as the Game of Life. It was proved that Life was capable of universal computation [7]. The proof of universality uses what in the jargon of CAs are known as gliders, glider guns,

and eaters, that is, structures to carry and manipulate information through the system. By combining such emergent propagating structures, logic gates and circuits can be simulated.

Langton's ant [8] is a two-dimensional Turing machine with 2 symbols and 4 states following a set of very simple rules. If the machine head is on a black square, it turns 90° right and moves forward one unit. If the head is on a white square, it turns 90° left and moves forward one unit. When the head leaves a square, it prints the opposite color. In [9], a very simple construction is presented that proves that Langton's ant is also capable of universal computation.

But an exhaustive exploration of one-dimensional elementary CAs (ECAs)—by most standards the simplest possible CAs—was undertaken in [2]. The rule with number 110 (and equivalent rules: 124, 137, and 193) in Wolfram's numbering scheme, which present the characteristic particle-like structures, turned out to be capable of universal computation [2, 10]. Rule 110 can be set up with initial configurations that have signals transmitted in the form of collisions of particle-like dynamical structures, simulating a variant of a tag system, another rewriting system capable of universal computation.

The proofs of universality for all these systems imply that their dynamics are unpredictable. The notion of universality implies the existence of undecidable problems related to most questions concerning these machines. Questions related to these simple dynamical systems cannot therefore be algorithmically answered. Because of this, undecidability is a measure of the unpredictability of a system associated with its dynamical behavior.

1.3 Quantitative Measures of Qualitative Behavior

Definition 1. [11–13] $K_U(s) = \min \{|p|, U(p) = s\}$ where $|p|$ is the length of p measured in bits with U , a universal Turing machine.

A measure of complexity is derived by combining the algorithmic complexity describing a system and the time it takes to produce a string. Bennett's concept of logical depth [4, 14] is a complexity measure capturing the structure of a string defined by the time that a Turing machine takes to reproduce the string from its (near) shortest description.

Definition 2. The logical depth D of a string is given by $D(s) = \min \{t(p) : (|p| < |p_i|) \vee U(p) = s\}$.

According to this measure, the longer it takes, the more complex the string. Complex objects are therefore those that can be seen as “containing internal evidence of a nontrivial causal history.” Bennett provides a careful elaboration [4] of the notion of logical depth, taking into account near-shortest programs as well as the shortest ones.

2. Compression-Based Phase Transition Coefficient

A measure based on the asymptotic direction change of the compressed evolution size of a system for different initial configurations (following a proposed Gray-code enumeration of initial configurations) was presented in [1]. It gauged the resiliency or sensitivity of a system in relation to its initial conditions. This phase transition coefficient led to an interesting characterization and classification of systems, which when applied to ECAs yielded exactly Wolfram's four classes of system behavior with no human intervention. The coefficient works by compressing the changes of the different evolutions through time, normalized by evolution space, and it is rooted in the concept of algorithmic complexity, since it is an upper bound of the algorithmic complexity of a string. The more compressed a string, the less algorithmically complex it is.

Let the characteristic exponent c_n^t be defined as the mean of the absolute values of the differences of the compressed lengths of the outputs of a system M running over the initial segment of initial conditions i_j . $j = \{1, \dots, n\}$ follows the numbering scheme devised in [1] based on a Gray-code optimal enumeration scheme, running for t steps in intervals of n .

Definition 3.

$$c_n^t = \frac{|C(M_t(i_1)) - C(M_t(i_2))| + \dots + |C(M_t(i_{n-1})) - C(M_t(i_n))|}{t(n-1)}.$$

Definition 4. Let C denote the transition coefficient of a system U defined as $C(U) = f'(S_c)$. The derivative of the line fits the sequence S_c by finding the least-squares as described in [1] with $S_c = S(c_i^n)$ for a fixed n and t .

The value $C(U)$, based on the phase transition coefficient, is a stable indicator of the degree of qualitative dynamical change of a system U . The larger the derivative, the greater the change. According to C , rules such as 0 and 30 appear close to each other both because they remain the same despite the change of initial conditions and because their evolution cannot be perturbed. The measure indicates that rules like 0 or 30 are also incapable of or inefficient at transmitting any information, given that they do not react to changes in the input of the system. Odd as it may seem, this is because there is no change in the qualitative behavior of these CAs when feeding them with different inputs, regardless of how different the inputs may be. Rule 0 remains entirely blank while rule 30 remains mostly random-looking, with no apparent emergent coherent propagating structures (other than the regular and linear pattern on one of the sides).

On the other hand, rules such as 122 and 89 appear next to each other as the most sensitive to initial conditions, because as the investi-

gation proves, they are both highly sensitive to initial conditions and present phase transitions that dramatically change their qualitative behavior depending on the initial configuration. This means that rules 122 and 89 can be more successfully used to transmit information from the input to the output.

■ 2.1 Connecting Dynamic Behavior and Turing Universality

Evidently, if a system is completely predictable and therefore dynamically trivial, it is decidable, and therefore not Turing universal. Rule 110 should therefore not be very predictable according to the phase transition measure, but at the same time we can expect it to be versatile enough to produce the variety needed to behave as a universal. Rule 110 is one rule about which the author's own phase transition classification says that, despite showing some sensitivity, also shows some stability. It can be said with some degree of certainty how it will look (and behave) for certain steps and certain initial configurations, unlike those at the top.

This is acknowledged by Wolfram himself when discussing rule 54 ([2] page 697): “It could be that if one went just a little further in looking at initial conditions one would see more complicated behavior. And it could be that even the structures shown above can be combined to produce all the richness that is needed for universality. But it could also be that whatever one does rule 54 will always in the end just show purely repetitive or nested behavior—which cannot on its own support universality.”

For every CA rule, there is a definite (often undecidable) answer to the question of whether or not it is capable of universal computation (or in reachability terms, whether a CA will evolve into a certain configuration). The question only makes sense if the evolution of a CA depends on its initial configuration. No rule can be universal if it fixes the initial configuration once and for all; there would be no way to input an instruction and carry out an arbitrary computation.

An obvious feature of universal systems is that they need to be capable of carrying information by reflecting changes made to the input and transmitted to the output. In attempting to determine whether a system is capable of reaching universal computation, it may be asked whether a system is capable of some minimal versatility in the first place, and how efficiently it can transmit information. And this is what the phase transition measures—it indicates how well a system manages to respond to an input. Obviously, a system such as rule 0 or rule 255, which does not change regardless of the input, is trivially decidable. But a universal system should be capable of reaction to external manipulation (the input to the system) in order to behave as a universal system, that is, to be capable of simulating and reaching the output of any other universal system.

Conjecture 1. Let U be a machine capable of (efficient) universal behavior. Then $C(U) > 0$.

Conjecture 1 is one-way only, meaning that it states that an efficient universal system should be equipped with these dynamical properties, but the converse does not necessarily hold, since having a large transition coefficient by no means implies that the system will behave with the freedom required for Turing universality. A case in point is rule 22, which, despite having the largest transition coefficient, seems restricted to a small number of possible evolutions.

2.2 Evidence and Discussion of a Qualitative Characterization

The conjecture is based on the following observations:

1. The phase transition coefficient provides information on the ability of a system to react to external stimuli.
2. Universal systems are (efficient) information processors capable of carrying and transmitting information.
3. Trivial systems and random-looking systems are incapable of transmitting information.
4. Trivial systems have negative C values, close to zero.
5. Rules such as 110, proven to be universal, and rule 54 (suspected to be universal; see [2] page 697) turn out to be classified next to each other with a positive transition coefficient.

The capacity for universal behavior implies that a system is capable of being programmed and is therefore reactive to external input. It is no surprise that universal systems should be capable of responding to their input and doing so succinctly, if the systems in question are efficient universal systems. If the system is incapable of reacting to any input or if the output is predictable (decidable) for any input, the system cannot be universal.

Values for the subclass of CAs referred to as elementary (the simplest one-dimensional CAs) have been calculated and published in [1]. We will refrain from evaluations of C to avoid distracting the reader with numerical approximations that may detract from our larger goal. The aim is to propose some basics of a behavioral characterization of computational universality. Figures 1 and 2 are examples of systems with different degrees of “programmability” related to their qualitative behavior.



Figure 1. ECA rule 4 is a kind of program filter that only transfers bits in isolation (i.e., when its neighbors are both white). It is clear that some very limited computations can be performed with this automaton.

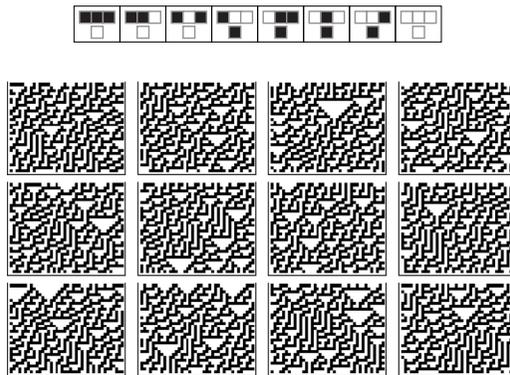


Figure 2. It is an open question whether ECA rule 30 can be programmed to perform computations. Its *C* value is low, meaning that it is not efficient for transferring information because it always behaves in the same fashion—to randomly.

For example, rules such as 0 do not produce different configurations relative to variant initial configurations. No matter how the initial condition is changed, there is no way to make it produce something other than what it computes for every other initial configuration. These trivial ECA rules are automatically ruled out, particularly the most simple among them that cannot usually be ruled out as candidates for universal behavior. This is because even if they look trivial for certain initial configurations, they could still be capable of the necessary versatility and eventually be programmed in light of the

space of all possible inputs for which they may be sensitive. The foundations of Conjecture 1 and the conjecture itself are consistent with all these observations, but it is most meaningful for systems that are believed to be of great complexity but are usually not believed to be malleable enough to be programmed as universal systems, such as with rule 30. If the conjecture is true, $C(U)$ may not only rule out systems that intuition strongly suggests are unable to behave as universals, but it would also indicate that random-looking systems such as rule 30 are not capable of universal computation because they are incapable of carrying information. In this sense, the measure may also be a characterization of the practical randomness of a system in terms of efficient information transmission.

Rule 110, however, has a positive C value, meaning it is efficient at carrying information from its input through the output, and that it can be programmed to perform computations. C is compatible with the fact that it has been proven that rule 110 is capable of universal computation.

A universal computer would therefore have a nonzero C limit value. C also captures some of the universal computational efficiency of the computer in that it captures not only whether it is capable of reacting to the input and transferring information through its evolution, but also the rate at which it does so. C is an index of both capability in principle and ability in practice. A nonzero C means that there is a way to codify a program to make the system behave (efficiently) in one fashion or another, that is, to be programmable. Something that is not programmable cannot therefore be taken to be a computer.

In [15], Margolus asserts that reversible cellular automata (RCAs) can actually be used as computer models embodying discrete analogues of classical notions in physics such as space, time, locality, and microscopic reversibility. He suggests that one way to show that a given rule can exhibit complicated behavior (and eventually universality) is to show (as has been done with the Game of Life [6] and rule 110 [2, 10]) that “in the corresponding ‘world’ it is possible to have computers” starting these automata with the appropriate initial states, with digits acting as signals moving about and interacting with each other to, for example, implement a logical gate for digital computation.

Conjecture 1 also seems to be in agreement with Wolfram’s beliefs concerning rule 30, which according to his PCE [2] may be computationally universal and still be impossible to control so as to be able to perform a computation (something that Wolfram has himself suggested [2]).

RCAs are interesting because they allow information to propagate, and in some sense they can be thought of as perfect computers—indeed in the sense that matters to us. If an RCA is started from a nonuniformly random initial state, the RCA evolves, but because it cannot get simpler than its initial condition (for the same reason given for the random state), it can only get more complicated, producing a

computational history that is reversible and can only lead to an increase in entropy.

3. On the Possible Computational Power of Busy Beaver Machines

3.1 Busy Beaver Machines

Rado [16] also studies the behavior of a special kind of one-tape n -state deterministic Turing machine, one that starts with a blank tape, writes more symbols that are not blanks than any other n -state Turing machine, and halts.

Note that we use $(n, 2)$ to denote the class (or space) of all n -state 2-symbol Turing machines (with the halting state not included among the n states).

Definition 5. [16] If σ_T is the number of 1s on the tape of a Turing machine T upon halting, then $\Sigma(n) = \max\{\sigma_T : T \in (n, 2) T(n) \text{ halts}\}$. If t_T is the number of steps that a machine T takes upon halting, then $S(n) = \max\{t_T : T \in (n, 2) T(n) \text{ halts}\}$.

$\Sigma(n)$ and $S(n)$ are noncomputable functions by reduction to the halting problem. Yet values are known for $(n, 2)$ with $n \leq 4$.

The busy beaver problem lies at the heart of what may be seen as a paradox. While a busy beaver machine of n states can be thought of as having maximal sophistication compared to all n state Turing machines regarding the number of steps and printed symbols, busy beaver machines can be extremely easily defined. The definition of busy beaver machines describes an infinite set of Turing machines characterized by a particular behavior—the attribute of printing more symbols that are not blanks on the tape before halting or having the longest runtime among all Turing machines of the same size (number of states).

Bennett's logical depth measure is relevant in characterizing the complexity of an n -state busy beaver machine both in terms of size (fixed among all n -state machines) and in terms of the behavior that characterizes this type of machine. It follows from Rado's definitions and Bennett's concept of logical depth that busy beavers are the deepest machines, provided that they are the ones with the longest history producing a string.

Yet a busy beaver is required to halt. When running for the longest time or writing the largest number of symbols that are not blanks, $bb(n)$ has to be clever enough to make wise use of its resources and still save a rule to halt. These facts may suggest the following conjectures, also in connection with the dynamic behavior of a set of simply described machines with universal behavior.

Conjecture 2.

1. Strong version: For all $n > 2$, $bb(n)$ is capable of universal computation.
2. Sparse version: For some n , $bb(n)$ is capable of universal computation.
3. Weak version: For all $n > 2$, $bb(n)$ is capable of (weak) universal computation.
4. Weakest version: For some n , $bb(n)$ is capable of (weak) universal computation.

It is known that no 2-state 2-symbol Turing machines can be universal. Remember, however, that $bb(n)$ as defined by Rado [16] is a Turing machine with n states plus a special halting state. So $bb(n)$ is actually a 3-state 2-symbol machine in which one state is specially reserved for halting only. By letting $bb(n)$ be a weak universal machine, initial tape configurations other than those filled with just a single symbol are allowed (usually called a blank tape, but blankness is a symbol in itself), but with initial configurations simple enough so that the computation is guaranteed not to be performed before it is given already computed in the input encoding. In other words, $bb(n)$ is allowed in Conjectures 2.3 and 2.4 to start either from a periodic tape configuration or an infinite sequence of the type accepted by a regular ω -automaton [17].

3.2 Discussion of the Characterization

If any version of the conjectures except Conjecture 2.4 is true, the characterization would define a countable infinite set of universal Turing machines. Their proof may provide an interesting framework and a possible path for proving a whole set of Turing machines to be capable of universal computation on the basis of their common dynamical properties.

Because halting machines that always halt cannot be capable of unbounded computation, and therefore of universal Turing behavior, among the analytical tools necessary to demonstrate the universality of any of these systems are proofs that busy beavers are capable of avoiding the halting state. If it is proven that busy beavers always halt, that would amount to proving that they cannot be universal. But to disprove Conjectures 2.1 to 2.3 it can be simply proved that at least one busy beaver is not capable of a halting configuration. A study of this type is likely to be simplified for $bb(3)$ or $bb(4)$, for which busy beaver functions are known and Turing machines are small enough to be subjected to a thorough and potentially fruitful investigation in this regard. The investigation of the behavior of busy beaver machines for initial configurations other than blank tape indicates that these machines are capable of nontrivial behavior when not in the simplest initial configuration. As intuition would suggest, if they behave in a sophisticated fashion for the simplest initial condi-

tion, they may be expected to continue doing so for more complicated ones. In a future paper, we will explore the specific behavior of these machines.

The truth of the conjectures may not seem intuitively evident to all researchers, given that it is possible that these machines are only concerned with producing the largest numbers by using all resources at hand, regardless of whether they do so intelligently. However, the requirement to halt is, from our point of view, a suggestion that the machine has to use its resources intelligently enough in order to keep doing its job while saving a special configuration for the halting state.

Conjecture 2.4 implies that being a busy beaver machine is not a characterization of the computational power of this easily describable set of countable infinite machines. But one intuition suggesting the truth about these conjectures is that it is easier to find a machine capable of halting and performing unbounded computations for a Turing machine if the machine already halts after performing a sophisticated calculation than it is to find a machine showing sophisticated behavior whose previous characteristic was simply to halt. This claim can actually be quantified, given that the number of Turing machines that halt after $t = n$ for increasing values of n decreases exponentially [18, 19]. In other words, if a machine capable of halting is chosen by chance, there is an exponentially increasing chance of finding that it will halt sooner rather than later, meaning that most of these machines will behave trivially because they will not have enough time to do anything interesting before halting.

We have no positive proof of any version of these conjectures, and much more work remains to be done on the dynamical behavior of these systems. But Conjectures 1 and 2 lead us to the final conjecture.

Conjecture 3. $C(bb(n)) > 0$.

4. Concluding Remarks

The first conjecture relates computational universality to the capacity of a computational system to transfer information from the input to the output and reflect the changes in the evolution of the system when starting out from different initial configurations. We established that the property of having a large phase transition coefficient seems necessary. On the other hand, a universal system seems to be capable of manifesting an abundance of possible evolutions and reacting to different initial configurations in order to (efficiently) behave universally.

A second conjecture concerning the possible universality of a kind of well-defined infinite set of abstract busy beaver Turing machines was introduced—also in terms of a measure of complexity related to algorithmic complexity and the dynamic behavior of these machines having a particular common characterization. The third conjecture relates Conjectures 1 and 2.

These conjectures will be the subject of further study in a paper to follow this one. We would like to see the conjectures proved or disproved, but underlying the conjectures are many other interesting questions relating to the size, behavior, and complexity of computing machines. It would be interesting, for example, to find out whether there is a polynomial (or exponential) trade-off between program size and the concept of simulating a process.

References

- [1] H. Zenil, "Compression-Based Investigation of the Dynamical Properties of Cellular Automata and Other Systems," *Complex Systems*, 19(1), 2010 pp. 1–28.
- [2] S. Wolfram, *A New Kind of Science*, Champaign, IL: Wolfram Media, Inc., 2002.
- [3] H. Zenil. "The Shortest Universal Machine Implementation Contest: FAQs." (Mar 7, 2008)
<http://www.mathrix.org/experimentalAIT/TuringMachine.html>.
- [4] C. H. Bennett, "Logical Depth and Physical Complexity," in *The Universal Turing Machine: A Half-Century Survey* (R. Herken, ed.), Oxford: Oxford University Press, 1988 pp. 227–257.
- [5] S. Wolfram, "Twenty Problems in the Theory of Cellular Automata," *Physica Scripta*, T9, 1985 pp. 170–183.
doi:10.1088/0031-8949/1985/T9/029.
- [6] M. Gardner, "Mathematical Games: The Fantastic Combinations of John Conway's New Solitaire Game 'Life'," *Scientific American*, 223(4), 1970 pp. 120–123. doi:10.1038/scientificamerican0169-116.
- [7] E. R. Berlekamp, J. H. Conway, and R. K. Guy, "What Is Life?," chapter 25 in *Winning Ways for Your Mathematical Plays*, Vol. 2, New York: Academic Press, 1982.
- [8] C. Langton, "Studying Artificial Life with Cellular Automata," *Physica D*, 22(1–3), 1986 pp. 120–149. doi:10.1016/0167-2789(86)90237-X.
- [9] A. Gajardo, A. Moreira, and E. Goles, "Complexity of Langton's Ant," *Discrete Applied Mathematics*, 117(1–3), 2002 pp. 41–50.
- [10] M. Cook, "Universality in Elementary Cellular Automata," *Complex Systems*, 15(1), 2004 pp. 1–40.
- [11] A. N. Kolmogorov, "Three Approaches to the Quantitative Definition of Information," *Problems of Information and Transmission*, 1(1), 1965 pp. 1–7. doi:10.1080/00207166808803030.
- [12] G. J. Chaitin, *Algorithmic Information Theory*, Cambridge: Cambridge University Press, 1987.
- [13] L. Levin, "Universal Search Problems," *Problems of Information Transmission*, 9(3), 1973 pp. 265–266.

- [14] C. H. Bennett, “How to Define Complexity in Physics and Why,” in *Complexity, Entropy, and the Physics of Information: The Proceedings of the 1989 Workshop on Complexity, Entropy, and the Physics of Information*, Santa Fe, NM (W. H. Zurek, ed.), Redwood City, CA: Addison-Wesley, 1990 pp. 137–148.
- [15] N. Margolus, “Physics-Like Models of Computation,” *Physica D*, **10**(1–2), 1984 pp. 81–95. doi:10.1016/0167-2789(84)90252-5.
- [16] T. Rado, “On Non-Computable Functions,” *The Bell System Technical Journal*, **41**(3), 1962 pp. 877–884.
- [17] T. Wolfgang, “Automata on Infinite Objects,” in *Handbook of Theoretical Computer Science* (J. van Leeuwen, ed.), Vol. B, Cambridge, MA: MIT Press, 1990 pp. 133–191.
- [18] C. S. Calude and M. A. Stay, “Most Programs Stop Quickly or Never Halt,” *Advances in Applied Mathematics*, **40**(3), 2008 pp. 295–308. doi:10.1016/j.aam.2007.01.001.
- [19] H. Zenil, “From Computer Runtimes to the Length of Proofs: On an Algorithmic Probabilistic Application to Waiting Times in Automatic Theorem Proving,” in *Computation, Physics and Beyond: International Workshop on Theoretical Computer Science (WTCS12)*, Auckland, New Zealand (M. J. Dinneen, B. Khoussainov, and A. Nies, eds.), New York: Springer, 2012 pp. 224–240.