Philosophy of Science (2022), **89**, 761–783
doi:10.1017/psa.2022.5

PSA PHILOSOPHY OF SCIENCE ASSOCIATION
PHILOSOPHY OF SCIENCE

ARTICLE

Setting the Demons Loose: Computational Irreducibility Does Not Guarantee Unpredictability or Emergence

Hamed Tabatabaei Ghomi

Department of History and Philosophy of Science, University of Cambridge, Cambridge, UK
Email: ht396@cam.ac.uk

(Received 14 January 2021; revised 07 August 2021; accepted 20 October 2021; first published online 11 February 2022)

Abstract

A phenomenon resulting from a computationally irreducible (or computationally incompressible) process is supposedly unpredictable except via simulation. This notion of unpredictability has been deployed to formulate recent accounts of computational emergence. Via a technical analysis, I show that computational irreducibility can establish the impossibility of prediction only with respect to maximum standards of precision. By articulating the graded nature of prediction, I show that unpredictability to maximum standards is not equivalent to being unpredictable in general. I conclude that computational irreducibility fails to fulfill its assigned philosophical roles in theories of computational emergence.

1. Introduction

Predictability is a common area of interest and investigation for both scientists and philosophers. It is not surprising, therefore, that some scientific theories on predictability have made their way into philosophical discussions. In particular, some philosophers have used computer science theories to formulate accounts of emergence that inform critical debates such as the nature of mind, the autonomy of the special sciences, or the origin of biological novelty (Bedau 1997; Huneman 2008, 2012; Humphreys 2016). In this paper, I use one of these computational theories, Wolfram's computational irreducibility (Wolfram 2002), or as some call it computational incompressibility (Bedau 2008; Huneman 2008), as a foil to show how these theories fail to fulfill some of their assigned philosophical roles.

There are two major types of emergence discussed in the literature, weak and strong, with two corresponding types of unpredictability. The weakly emergent are reducible to their fundamental bases and therefore are unpredictable only in practice, while the strongly emergent are irreducible and therefore are unpredictable in principle. There is a third type of emergence discussed in the recent literature that sits somewhere between weak and strong emergence. According to computational

emergence, or as I will call it, ontological weak emergence, emergent phenomena are reducible to their bases, yet because of their specific computational characters, namely their computational irreducibility, they are, as Bedau puts it, unpredictable in practice in principle.

Wolfram (2002) claims that from the computational perspective, the outcomes of computationally irreducible processes are unpredictable for any observer. Philosophers such as Bedau (1997) and Huneman (2008) have adopted this view to formulate ontological weak emergence, according to which emergent phenomena are unpredictable for any observer because of the computational irreducibility of the path leading to those phenomena. These computational accounts of emergence supposedly have the best of the two worlds of weak and strong emergence, and are doubly attractive for the scientifically minded. They retain the familiar scientific metaphysics of weak emergence while they underpin the unpredictability of the emergent with an ontological and scientifically backed explanation.

However, I show that computational irreducibility, no matter how it's interpreted, guarantees unpredictability only with maximal standards of accuracy and precision, while the scientifically and practically relevant concept of predictability is graded and submaximal. This creates a dilemma for supporters of computational emergence; they cannot have their ontological cake and eat it, too. If they cite computational irreducibility as their ontological guarantee of unpredictability of the emergent, they have to stick to the maximal standards of prediction. The unhappy consequence, however, would be that all natural phenomena would trivially turn out to be unpredictable and "emergent," and computational emergence would be indiscriminatory, trivial, and irrelevant to the intuitive concept of emergence and the practical puzzles it creates.

On the other hand, if they ease the standards of prediction, computational irreducibility can no longer guarantee unpredictability and computational formulations of emergence will be sullied by subjectivity and arbitrariness. Consequently, computational ontological weak emergence would not be much different or superior compared to simpler and more flexible non-ontological accounts of weak emergence. If successful, the arguments will take ontological weak emergence off the table, and we will go back to the choice between weak and strong emergence.

2. Computational irreducibility

Wolfram describes the causal or computational path leading to a phenomenon as an algorithm that computes that phenomenon. The original algorithm generating a phenomenon is computationally irreducible if it is the shortest possible path to derive that phenomenon (Wolfram 2002). As there is no alternative pathway to compute the resulting phenomenon faster than the original algorithm, the resulting phenomenon is unpredictable (Figure 1).

Wolfram presents the concept of computational irreducibility in the context of cellular automata, although the concept can be extended beyond that context. Cellular automata (CA) are lattices of cells called the *cell-space*, where each cell stores a value such as a color. Starting from some initial values, the values of cells are updated in consecutive timesteps by some *updating rules*. At each timestep, the updating rule looks at a cell's own value and its neighboring cells' values and

https://doi.org/10.1017/psa.2022.5 Published online by Cambridge University Press

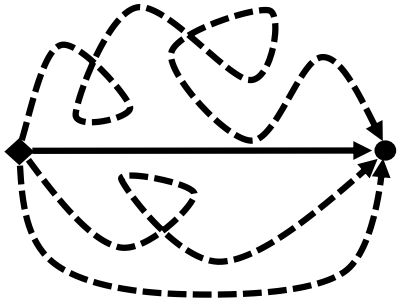

Figure 1. A computationally irreducible path to a phenomenon (solid line) is the shortest path to that phenomenon.

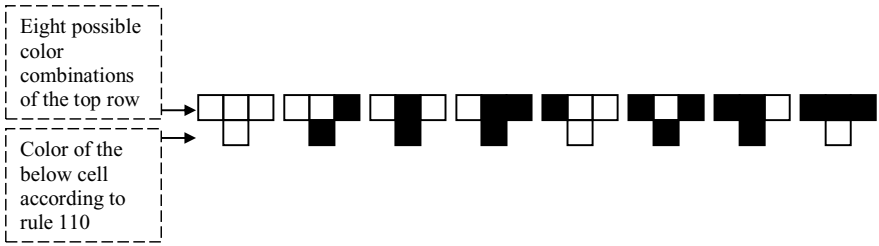

Figure 2. Overview of Wolfram's CA set up. Rule 110 is given as an example.

determines the cell's value in the next timestep. Starting from some initial values and following the updating rules, some general patterns of values form on the cell-space as time goes by. The value of any single cell in each step is the *micro-dynamics* of a CA, and the overall patterns formed on the cell-space are the *macro-states*. Various combinations of cell shapes, dimensions of cell-space, cell values, and updating rules result in different CA (Charbonneau 2017; Berto and Tagliabue 2017).

Wolfram works with a particular setup of one-dimensional CA. Consider a chessboard with all white cells. Start by coloring some of the first top row cells black. In each iteration, move one row down and color cells based on the color of their neighboring cells on the immediately above row following a set of updating rules. Each cell has three neighbors in the top row, and one, two or three of these neighbors might be black, allowing eight different color combinations (Figure 2). An updating rule states whether a cell will be colored black or not in each of these eight possible combinations. This means that we can have $2^8 = 256$ different updating rules. These rules are numbered 0 to 255. Starting from some black and white cell configuration on the first row, different rules result in evolution of different general patterns on CA.

Wolfram categorizes these 256 rules into four classes (Figure 3). Class one comprises of those rules under which CA converges into a uniform final state, for example, all cells become black (e.g., rule 250). Class two are those rules that result in some simple periodic behavior (e.g., rule 108), and class three are those that lead to random patterns (e.g., rule 90). Class four generates patterns that are a combination of randomness and periodicity (e.g., rule 110).

https://doi.org/10.1017/psa.2022.5 Published online by Cambridge University Press

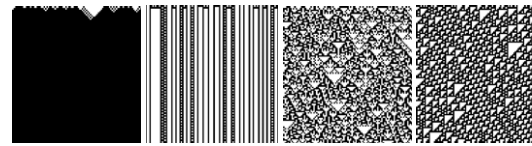

Figure 3. Examples of CA in each of Wolfram's four classes.

Out of these four classes, the outcome of classes one and two are predictable using simple constant or oscillating predicting models. Using such predicting models, there is no need to simulate a class one or two CA in order to know its outcome. But when it comes to classes three and four, Wolfram claims that there are no predicting models and one has no alternative but to simulate the CA and watch what happens. Unlike the cases of classes one and two, there are no shortcuts, and the fastest way to know the outcome of the CA is going through the CA itself. These CA, therefore, are computationally irreducible (Wolfram 2002).

Computational irreducibility is not limited to CA. Any algorithmic process would be computationally irreducible if the computationally most efficient way to know its outcome is running the algorithmic process itself. This is a mathematical hard lower bound and no observer whatsoever, not even a Laplacian demon, can pass it. The output of such an algorithmic process will be unpredictable because there is no way to compute its outcome faster than running the algorithm itself. In Wolfram's words (2002, 739): "Whenever computational irreducibility exists in a system it means that in effect there can be no way to predict how the system will behave except by going through as many steps of computation as the evolution of the system itself."

The above apparently clear statement is in fact utterly ambiguous. Wolfram does not precisely clarify what he means by "as many steps of computation" and does not give a formal proof for his claim. Therefore, it remains unclear exactly in *what* sense and *why* a computationally irreducible process is irreducible after all. There are a few options to explain these *what* and *why*. I explore these options below and examine the type of unpredictability each can guarantee. The exploration will show that computational irreducibility can guarantee unpredictability only for *all cases*, in *infinite time*, or with *infinite precision*.

2.1 Computational irreducibility as a case of the halting problem

Wolfram deduces computational irreducibility from his *principle of computational equivalence*. This principle has multiple components, but the most important for the present discussion is the conjecture that any process, in nature or a computer, that does not converge to a constant or to an oscillating pattern is a *universal Turing machine* (Mitchell 2009).

Turing machines are models of computers initially introduced by Alan Turing (1937) and now widely used in computer science. A simple Turing machine is composed of an infinite tape of individual cells as its memory and a head that points to one of these cells at a time. The head can read the cell it is pointing to, write on it, and move to the next neighboring cell on either left or right. The machine can be in

https://doi.org/10.1017/psa.2022.5 Published online by Cambridge University Press

one of a set of predefined states. The behavior of the head is dictated by a transition function based on the current state of the machine and the symbol read on the cell. The transition function determines what the head should write on the cell, the direction the head should move to, and the state to which the machine should transit. A Turing machine begins at a "start" state and halts when it enters one of the "final" states. When the machine enters one of the final states, the computation finishes. But it is also possible that the machine never enters a final state and infinitely iterates over a loop of nonfinal states.

According to the Church-Turing thesis, any computable function is computable by some Turing machine. There are "universal" Turing machines that can simulate any other Turing machine and, therefore, can compute any computable function. Cook has shown that Rule 110 CA is a universal Turing machine (Aaronson 2002). This means that given the right initial conditions, a rule 110 CA can compute any computable function. Figuring out the right initial conditions for computing different functions with this CA is extremely hard. But it is in principle possible.

Although only a very limited number of CA are demonstrated to be universal Turing machines (Aaronson 2002; Berlekamp, Conway, and Guy 1982; Rendell 2011), Wolfram conjectures that any CA that does not converge to a constant or an oscillating pattern is a universal Turing machine. And because he sees natural processes as CA, he extends this claim to natural processes as well. Any natural process that does not result in a constant or oscillating behavior, therefore, is a universal Turing machine, or so Wolfram claims. Both the initial conjecture and the extension to natural processes are open to objection. Nonetheless, let us grant both the conjecture and the extension and see where that would lead us.

Being a universal Turing machine subjects CA to the *halting problem*. The halting problem states that there is no single algorithm that can predict for any arbitrary input whether a universal Turing machine will halt. This can be one interpretation of computational irreducibility. Based on this interpretation, some CA are computationally irreducible and hence unpredictable because there is no general algorithm to predict if they will halt given some input. The only way to know whether they halt is to run the CA themselves (Ilachinski 2001). For example, there is no single algorithm that can predict for any arbitrary initial pattern if Rule 110 will halt or continue to iterate indefinitely.

This unpredictability is not because of some technological limitation or lack of mathematical ingenuity. The halting problem dictates that halting or not halting of the computationally irreducible CA is unpredictable *in principle*. No technological breakthrough nor mathematical genius can ever devise a path around the halting problem (Yanofsky 2016). Even a Laplacian demon would be stalled.

The conclusion drawn from the halting problem, however, is very restricted. The halting problem shows that no single algorithm can predict halting or not halting for all computations and for any arbitrary input. But the halting problem does not show that we cannot have multiple predicting algorithms that each correctly predicts halting or not halting for a broad subset of computations and inputs. Being subject to the halting problem is completely consistent with being predictable in many instances. The halting problem only shows that we cannot predict the halting or not halting for all and every case.

https://doi.org/10.1017/psa.2022.5 Published online by Cambridge University Press

In other words, the halting problem shows that there cannot be one single "Grand Algorithm of Everything" that predicts the fate of every natural process for all inputs. In this way, it seems that there might be a path from the halting problem to the impossibility of a "Theory of Everything." But the halting problem cannot show that particular natural processes are necessarily unpredictable by any theory. Scientists can appreciate the halting problem and yet go back to their benches and blackboards to make predictions.

I believe the recourse to the halting problem is the closest interpretation of computational irreducibility to what Wolfram has presented in his works. We saw that the unpredictability we can conclude from this interpretation is very restricted. But there are other ways to interpret computational irreducibility and those might be more successful in securing unpredictability. Let us explore those alternatives.

2.2 Algorithmic computational irreducibility in terms of algorithmic time complexity

Another way to understand Wolfram's irreducibility is to define it in terms of *algorithmic time complexity* (Zwirn and Delahaye 2013; Rucker 2003; Zwirn 2013; Huneman 2008). I explore a number of different versions of this approach in this section. But at their core and on a very rough sketch, all of those define a computationally irreducible algorithm as the one with the lowest (=best) algorithmic time complexity. It turns out that these formulations can guarantee unpredictability only over long times, and with infinite accuracy and precision.

Algorithmic time complexity is the computer science currency of speed and efficiency. In simple terms, it shows how the time needed for an algorithm to solve a problem scales with some measure of the input size. The time is estimated by the number of the elementary operations that an algorithm goes through to give the solution, and that number varies as a function of the input size. Time complexity is the *order* of this function which is expressed by notations such as the big O notation and is used to rank algorithms according to their efficiency.[1] The order shows how the number of operations scales with the input size. The lower the order, the more efficient the algorithm. Although time complexity is initially defined for algorithms, computer science usually associates time complexity with problems and not algorithms. The time complexity associated with a problem is the time complexity of the most efficient algorithm to solve it. Here, however, we need to zoom in, and discuss the time complexity of individual algorithms.

Here I devise a simple and familiar programming language to write the pseudocodes. INPUT (X) means receiving an input and assigning its value to X. FOR (X) means repeating what is contained in the FOR-block X times. The FOR-block is determined by the brackets, the first of which is placed on the line immediately following the FOR command. OUTPUT (X) means finishing program and returning X as the output. Let us look at an example. Suppose you want an algorithm to compute the number of bacteria after n generations starting from a single

[1] Here I use the time complexity notations very loosely based on the general idea behind them and not their precise formalism. I stick with the big O notation as it is a weaker requirement compared to the small o notation.

https://doi.org/10.1017/psa.2022.5 Published online by Cambridge University Press

bacterium. For simplicity, assume that no bacterium dies. In every generation, each bacterium divides and makes two daughter bacteria. This means that in each generation, one bacterium is added to the population per every bacterium in the previous generation. Here is the first algorithm (algorithm A) inspired by this observation:

Algorithm A:

```
INPUT (n)
current number of bacteria = 1
FOR (n)
        {
        number of bacteria in the previous generation = current number of bacteria
        FOR (number of bacteria in the previous generation)
                {
                current number of bacteria = current number of bacteria + 1
                }
        }
OUTPUT (current number of bacteria)
```

To determine the time complexity of this algorithm, we need to count the number of operations it performs for input n. The first FOR loop repeats the operations within its brackets for n generation. In each generation, the algorithm performs one addition per present bacteria. In the first generation, it does one addition, in the second, it does 2, in the third, it does 4, and so forth. The total number of additions performed by algorithm A, therefore, can be computed by the following formula:

$$1 + 2 + 4 + 8 + \ldots + 2^{n-1} = \sum_{i=0}^{n-1} 2^i = 2^n - 1$$

The function $2^n - 1$ is of order 2^n. Therefore, algorithm A has time complexity $O(2^n)$. It means that as the number of generations n increases, the number of operations needed to get the result, and consequently, the time spent to run the algorithm scales by the order of 2^n. This is an inefficient time complexity, and there are algorithms of lower time complexities to solve this problem. For example, noting that the number of bacteria doubles in each generation, one can write algorithm B:

https://doi.org/10.1017/psa.2022.5 Published online by Cambridge University Press

Algorithm B:

INPUT (n)

current number of bacteria = 1

FOR (n)

{

current number of bacteria = 2 × current number of bacteria

}

OUTPUT (current number of bacteria)

Algorithm B has time complexity $O(n)$, and this is significantly more efficient compared to algorithm A. If we understand computational reduction in terms of time complexity, then algorithm A is computationally *reducible* to algorithm B. We can use algorithm B to compute the result of algorithm A with a lower time complexity. And as algorithm B is more efficient than algorithm A, so the thought goes, we can predict the results of algorithm A using algorithm B.

There is an even more efficient algorithm to solve this problem. We know that starting from one bacterium, the number of bacteria after n generations equals 2^n. This translates to the following algorithm C:

Algorithm C:

INPUT (n)

current number of bacteria = 2^n

OUTPUT (current number of bacteria)

Note that 2^n is not an elementary operation and itself takes a few steps to compute. We have algorithms that compute 2^n with time complexity $O(\log(n))$.[2] Thus, algorithm C has time complexity $O(\log(n))$ that is a lower order compared to algorithm B's $O(n)$. Therefore, even algorithm B is reducible.

There is an important difference between algorithm C and the two previous algorithms A and B, and that difference has inspired one interpretation of computational irreducibility. Contrary to algorithms A and B, there is no looping over generations in algorithm C. Algorithms A and B walk through the generations of cell division one by one, and they compute the number of bacteria for generations 1 to n-1 before they

[2] For example, method of exponentiating by squaring.

https://doi.org/10.1017/psa.2022.5 Published online by Cambridge University Press

compute the number of bacteria in generation n. Algorithm C, on the other hand, does not go through this generation-by-generation path and jumps directly to generation n without computing the number of bacteria in the intervening generations. This is why algorithm C can be packaged as *a closed-form formula*, like a formula from physics:

$$\text{current number of bacteria} = 2^n$$

Rucker (2003) seems to interpret Wolfram's computational irreducibility as the impossibility of a closed-form formula. On this interpretation, a computationally irreducible algorithm is one the result of which cannot be computed by a closed-form formula and, therefore, one needs to necessarily walk through the steps of the algorithm to know its results. Imagine that algorithm C or any other closed-form formula were not possible to solve our bacteria problem, and the only way to compute the result was to do the calculations generation-by-generation. Then, on Rucker's interpretation, the bacteria problem would have been computationally irreducible.

The example of the three algorithms above, however, shows that the mere impossibility of a closed-form formula is not enough to guarantee irreducibility and unpredictability. Even if algorithm C were not possible, algorithms A and B could be ranked according to their time complexity with the idea that the result of the more efficient algorithm B can predict the results of the less efficient algorithm A.

Zwirn and Delahaye (2013) and Zwirn (2013) formalize a more detailed definition for algorithmic computational irreducibility in terms of time complexity. Here I try to avoid the mathematics of Zwirn's account as much as possible and emphasize the general idea. The mathematical reader is referred to Zwirn's original paper for the exact formalism. Assume an algorithmic process $F(n)$, that computes f_n by going through f_1 to f_{n-1}. Starting from some initial input value for f_1, in each time step i, f_i receives the output of f_{i-1} and computes an output, which in turn will be the input for f_{i+1}.[3] On this setup, Zwirn requires two conditions for computational irreducibility of $F(n)$. The first condition requires that given n as input, a computationally irreducible F should compute f_n with the best possible time complexity. There should be no alternative algorithmic process G that computes f_n with lower time complexity. The second condition requires that any other alternative algorithmic process G that computes the same result f_n by an alternative function g should necessarily do so by going through steps g_1 to g_{n-1} where each g_i is an approximation of its corresponding step f_i. By "approximation" Zwrin means that one can compute f_i from g_i by a very short computation.[4]

Assume, just for the sake of argument, that the above algorithms A and B are the only possible algorithms to solve the bacteria problem. This (wrong) assumption guarantees that B has the lowest possible time complexity. It also ensures that the single alternative algorithm (algorithm A) computes the number of bacteria in generation n by going through steps 1 to n-1, and its value in each step can be easily

[3] Zwirn takes all f_i to be the same function f, which runs iteratively, but I think this assumption is not necessary.

[4] Zwirn distinguishes *strong* computational irreducibility from computational irreducibility. The former satisfies the two conditions for any n, and the latter for infinitely many n. In this paper, I use computational irreducibility in the strong sense.

https://doi.org/10.1017/psa.2022.5 Published online by Cambridge University Press

transformed to the value computed by algorithm B (by B(n) = A(n)). B, therefore, would be computationally irreducible by Zwirn's definition.

Zwirn formulation has limited power to show unpredictability of the computationally irreducible. The reason is that having the same/lowest time complexity does not necessarily mean having the same/lowest absolute number of computations and the same/lowest run time in all conditions. For example, consider algorithm B′ that solves the bacteria problem mentioned above as follows:

Algorithm B′:

INPUT (n)

current number of bacteria = 1

FOR (n)

{

current number of bacteria = $\frac{100\times101}{\sum_{i=1}^{100} i}$×current number of bacteria

}

OUTPUT (current number of bacteria)

B′ has the same time complexity as algorithm B, but it has a higher absolute number of computations and therefore, it runs slower than B (Figure 4). The difference between B and B′ will eventually become insignificant as *n* increases to infinity. But in finite time, B can compute the results faster than B′ and, therefore, can predict its outcome. Even more, B′ has a better time complexity compared to algorithm A, but until n=10, A has a lower absolute number of computations and therefore, runs faster than B′ and can predict its outcome (Figure 4).

It is only when *n* increases toward infinity that the difference between algorithms of the same time complexity diminishes to insignificance and the algorithms with the best time complexity are guaranteed to be faster than the alternative algorithms with higher time complexities. Therefore, computational irreducibility as defined by Zwirn guarantees unpredictability only toward the infinite time. For shorter times, a computationally irreducible algorithm might be predictable. Unpredictability based on Zwirn's definition is also limited by the possibility of heuristic solutions that I discuss at the end of this section.

On another interpretation, a process is computationally irreducible if predicting its outcome is an NP-hard problem (Buss, Papadimitriou, and Tsitsiklis 1990; Huneman 2008). Roughly speaking, NP-hard problems are those for which we do not yet have a solution with polynomial time complexity ($O(n^{\alpha})$, $\alpha > 1$).[5] For any relatively large

[5] Buss et al. (1990) use space complexity rather than time complexity. The two types of complexity are closely connected, and as far as the arguments of this paper are concerned, there is no difference between the two.

https://doi.org/10.1017/psa.2022.5 Published online by Cambridge University Press

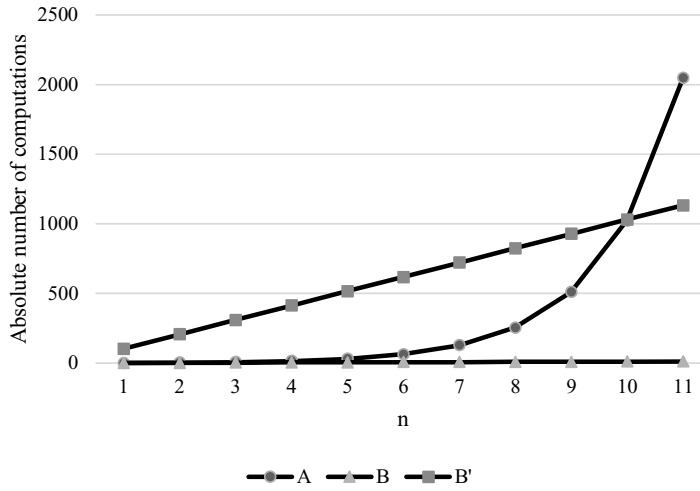

Figure 4. Comparison of the absolute number of computations in three algorithms. B and B' have the same time complexity. A has a higher time complexity compared to B and B'.

input size, it takes a prohibitively long time to solve a problem with a time complexity of higher than polynomial, and, therefore, NP-hard problems are practically unsolvable, or, as computer scientists call them, *intractable*. Intractability applies not just to us, the humans of the twenty-first century, but to any natural intelligence. For example, consider the Traveling Salesman, a classic NP-hard problem. The problem is finding the shortest path to visit each of n towns exactly once and come back to the starting town. A demon who can use all the atoms of the universe as processing units needs 10^{62} centuries to solve the Traveling Salesman problem for an input size of $n = 100$ (Yanofsky 2016). As an NP-hard problem becomes intractable relatively quickly, we do not need to go toward infinite time for the prediction to become practically impossible.

Nevertheless, being NP-hard is also restricted in what it guarantees to be unpredictable. It is only the exact solution for an NP-hard problem that is intractable, and an approximate solution might be computable in much less time. In fact, NP-hard problems constitute a good portion of practical problems we are dealing with from everyday life to advanced science. Assigning wedding guests to seats in a way that friends share a table but foes do not (Lewis and Carroll 2016), or constructing a phylogenetic tree (Habib and Stacho 2013) are some examples. Computer scientists may not be able to provide exact solutions to these problems, but they have devised plenty of approximate solutions that are both fast and acceptably accurate. It does not even take an intelligent computer scientist to find fast approximate solutions for NP-hard problems. Zhu et al. (2018) have shown that even as humble intelligence as an amoeba can find linear time approximate solutions for NP-hard problems.

Approximate solutions for NP-hard problems are examples of heuristic approaches that find approximate solutions for problems that are hard, or even impossible to solve. One particular approach that is specifically relevant to our discussion is coarse-graining of CA. Israeli and Goldenfeld (2006) have shown that computationally

https://doi.org/10.1017/psa.2022.5 Published online by Cambridge University Press

irreducible CA can be turned into reducible ones by coarse-graining. Coarse-graining of CA means combining some neighboring cells into one cell. The process results in a lower-resolution and an imprecise re-description of a CA pattern. Israeli and Goldenfeld show that the patterns generated by a computationally irreducible class 4 rule (e.g., Rule 110) may be re-described as patterns of another rule of a lower class after coarse-graining (e.g., Rule 0) that are not computationally irreducible. This means that computational irreducibility and its consequent unpredictability hold only for full precision CA. No matter how we define computational irreducibility, we lose it when we go to a coarse-grained view of CA.

In summary, different accounts of computational irreducibility based on algorithmic time complexity can guarantee unpredictability only for infinite time and infinite precision. Wolfram is aware of this problem, and he emphasizes this point in his previous papers. He writes (Wolfram 1984, 424): "The large time limit of the entropy for class 3 and 4 cellular automata would then, in general, be non-computable: bounds on it could be given, but there could be no finite procedure to compute it to arbitrary precision."

But do we need arbitrary precision for predictability? And how far into the future should this "large time limit" be? In the next section, I argue that to have a predictable phenomenon we do not need arbitrary precision and we do not need to predict until indefinitely long time limits.

3. Prediction is graded

We generally understand prediction as describing a phenomenon before observing it, and we talk about "successful" or "failed" predictions, and accordingly, we say something is "predictable" or "unpredictable." These expressions of success, failure, predictability, and unpredictability represent prediction as a black-and-white concept that is either successful or unsuccessful, possible or not possible. This language usage hides the fact that prediction extends over a spectrum, and it is almost always not black or white, but grey.

This graded nature of prediction is well reflected in the science of prediction. There are fields like weather or political forecast where the success of prediction has been subject of heavy investigation. The metrics that these fields use to gauge the success of prediction rank various predictions on a range between complete success and complete failure. An example is the Brier score, one of the most widely used of these metrics (Brier 1950). Brier originally formulated the score to measure the goodness of weather predictions, but later the score made its way to studies of socio-political predictions as well (Tetlock and Gardner 2016). Brier score has the following formula:

$$B = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{r}\left(p_{ij} - E_{ij}\right)^2$$

The index j corresponds to each possible outcome, and r shows the total number of possible outcomes. For example, it might rain ($j = 1$) or not rain ($j = 2$ and in this case, $r = 2$). The index i corresponds to an occasion of prediction, and n shows the total number of the predictions made. For example, we may predict whether it rains or not for ten days, and in that case $i = 1,\ 2,\ \ldots, 9,\ 10$ and $n = 10$ p_{ij} is the

https://doi.org/10.1017/psa.2022.5 Published online by Cambridge University Press

probability that the forecaster assigns to outcome j on occurrence i. E_{ij} indicates whether outcome j has happened on an occurrence i ($E_{ij} = 1$), or not ($E_{ij} = 0$). The best Brier score is zero, and the lower the Brier score, the better.

Suppose we have four weather forecasters. The Archangel who predicts all the rainy and not rainy days correctly and is always 100% confident about the predicted outcome. The Demon who mispredicts all days but is also 100% confident. The Bold forecaster who correctly predicts 60% of the rainy days, but she is 90% confident about her predictions. And the Cautious forecaster who also predicts 60% of rainy days correctly, but she is only 60% confident about her predictions. Table 1 shows the predictions of these four forecasters over ten consecutive rainy days and the corresponding Brier score.

The best score is zero, and it goes to Archangel. The worst is 2, and it goes to Demon. The interesting point is the score differentiates between Cautious and Bold, although both make the same predictions. The better score of Cautious compared to Bold implies that someone who predicts with 60% accuracy should be only 60% confident in her predictions. Bold gets a worse score for being over-confident. The confidence of a prediction is not necessarily a subjective perception in the eyes of the forecaster. The probability that a probabilistic natural law assigns to an outcome can replace the confidence a forecaster puts in her forecast. In this way, the Brier score can rank predictions of alternative probabilistic laws.

One might come up with other scores to measure the success of predictions of a law or a forecaster. For example, the formulation of the Brier score that I presented is suitable only for predictions of binary outcomes such as rain or no rain. But when the outcome can have a range of values, the above formulation of the Brier score would not be sufficient. We would need an alternative score that incorporates the precision of the prediction as well. For example, if the temperature is 14 Celsius, a prediction of 14.5 Celsius must get a better score than a prediction of 16 Celsius. The new score should somehow combine accuracy, certainty, and precision. Or one might come up with another score that looks also at how far in the future the forecaster can go. A forecaster who correctly predicts the weather a year from now should get a better score than a forecaster who predicts only tomorrow's weather. Or another score might incorporate the variety of climate types that a forecaster can predict. A forecaster who successfully predicts rainy days only for tropical rain forests would get a lower score compared to a forecaster who predicts rainy days in various geographic regions. The critical point is that no matter what criteria we add, all these alternative scores will put the success of a prediction on a spectrum.

Even the most successful scientific predictions fall short of the absolute ends of success spectrum for various reasons. For example, most of scientific predictions are never maximally certain. The reason is evident if the laws used for prediction are probabilistic. A probabilistic law predicts the outcome with a less than one probability and leaves the possibility that the predicted outcome does not happen. Some laws of science, such as some quantum mechanical laws, are explicitly probabilistic. But even some of the laws that look deterministic on their surface are in one way or another probabilistic at their core. For example, chemistry teaches us that carbon dioxide interacts with water and produces carbonic acid. But it is more accurate to say that carbon dioxide interacts with water and produces carbonic acid only if the molecules collide at the correct angle and with sufficient energy. And such a

https://doi.org/10.1017/psa.2022.5 Published online by Cambridge University Press

Table 1. Some predictions and their corresponding Brier scores

	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10	Brier score
Archangel	100% Rainy	100% Rainy	100% Rainy	100% Rainy	100% Rainy	100% Rainy	100% Rainy	100% Rainy	100% Rainy	100% Rainy	0
Demon	100% Sunny	100% Sunny	100% Sunny	100% Sunny	100% Sunny	100% Sunny	100% Sunny	100% Sunny	100% Sunny	100% Sunny	2
Cautious	60% Sunny	60% Rainy	60% Rainy	60% Rainy	60% Sunny	60% Rainy	60% Sunny	60% Rainy	60% Sunny	60% Rainy	0.48
Bold	90% Sunny	90% Rainy	90% Rainy	90% Rainy	90% Sunny	90% Rainy	90% Sunny	90% Rainy	90% Sunny	90% Rainy	0.66

https://doi.org/10.1017/psa.2022.5 Published online by Cambridge University Press

collision is a probabilistic event. Many laws of the special sciences are like this. They are stated with a deterministic tone, but a closer look shows that they are probabilistic at their core.

Scientific predictions are also not maximally accurate and precise because of the noise inherent in any measurement and experiment. Part of this noise comes from the random fluctuations that are always there in the context of any experiment or measurement. And another part comes from the upper cap of precision and accuracy of the measurement devices. But there is also a third source of noise that comes from the limited precision of the values stored in computers. Regular computers have between seven to sixteen decimal digits of precision. Going to more accurate computers mitigates this type of noise but does not reduce it to zero and there is a hard upper cap on the precision of even the ultimate hypothetical super-computer (Davies 2004).

Scientific predictions also cannot look arbitrarily far into the future. For all the reasons discussed, scientific predicting models all have small systematic or nonsystematic errors and those cause problems for predictor simulations. Given long enough, small systematic deviations will result in humongous differences between the predictions and the actual outcome. In very long time periods, even nonsystematic errors might find the chance to accrue in one direction at some point in time, and significantly change the course of the outcome from then on. And both systematic and nonsystematic small deviations might significantly deviate the prediction from the actual outcome due to the butterfly effect. Different cases of prediction would be more or less robust over long periods of time and would successfully see farther or closer into the future. But no real-world prediction goes until infinite time.

Save the not-yet-found theory of everything, scientific models are also not supposed to predict everything, everywhere. All scientific models work within a limited scope and under certain assumptions. The wider the scope of a model and the fewer its assumptions, the better the model. But the scope never becomes all-encompassing and the assumptions never go to zero. Of course, some level of generality is duly expected from a scientific model. But the generality need not expand to every phenomenon and every situation.

In short, even the best predictions in science are never maximally accurate, precise, and certain, and they are not expected to predict until the infinite future. They are also not expected to predict everything, everywhere. These less than maximal standards apply even to the astonishingly accurate astrophysical predictions and the profoundly accurate scientific devices such as atomic clocks. Maximal standards are too much to ask even from the paradigm successful predictions.

We saw computational irreducibility guarantees impossibility of prediction only for *all cases*, in *infinite time*, or with *infinite precision, accuracy, and certainty*. The above discussions, however, show that lack of prediction under these maximal conditions is not equivalent to being unpredictable because maximal standards are not necessary for predictability. It follows that a theory like computational irreducibility that shows some phenomenon cannot be predicted with maximal standards does not show that the phenomenon is necessarily unpredictable.

Computational irreducibility, therefore, cannot provide a basis for unpredictability. We see the philosophical importance of this conclusion in more details in the next section in the context of computational accounts of emergence.

https://doi.org/10.1017/psa.2022.5 Published online by Cambridge University Press

4. Ontological weak emergence and computational irreducibility

An emergent phenomenon is a feature of a system in its entirety that looks unpredictable and novel given all that we know about the system's parts. We are confident that we know the micro-structure of a system well enough to explain and predict its systemic macro properties and behaviors. Yet, the system shows some macro-level features that we cannot predict or explain. There are different ways to explain this dichotomy between what we expect from the micro-structure and what we observe at the macro-level. One way is the approach suggested by Darley (1994) and Bedau (1997, 2008) and supported and expanded by Huneman (2008, 2012). I call this approach *ontological weak emergence*.[6] Due to the heavy reliance of this approach on computational concepts, it is also called *computational emergence* (Huneman 2008).

Ontological weak emergence sits somewhere between *strong* and *weak* emergence. Strong emergence suggests that emergent phenomena are ontologically distinct from their underlying micro-structure. The most famous examples are mental phenomena that, according to strong emergentists, are of a different nature from the neurological system underneath them (Chalmers 1996, 2008). Weak emergence, on the other hand, suggests that there is nothing ontologically special about emergent phenomena, and they look novel and unexpected simply due to our limited understanding of the underlying system. In this view, emergence is merely an epistemic perspective in the eyes of the beholder, and it will eventually dissolve as the beholder develops a better understanding of the system (Hempel and Oppenheim 1948; Chalmers 2008).[7]

Weak emergence has a special appeal due to the current popularity of physicalism, as it does not introduce any new ontological types and keeps the physicalist ontology clean and tidy. Weak emergence, however, does not explain the notorious persistence and prevalence of emergent phenomena in so many disciplines, from biology to sociology. The history of science shows that many emergent phenomena, such as almost any phenomenon of interest in biology, economics, and so on, still count as emergent despite significant advances in more fundamental sciences. The early supporters of weak emergence, such as Hempel and Oppenheim, had a different vision of the future. According to Hempel and Oppenheim (1948), during the course of scientific progress, phenomena are only transiently emergent because the theories and facts to explain them are yet to be discovered. Once the advances in science provide the necessary theories and facts, these will shed light on the phenomenon and dispel the apparent magic of emergence. Although there would be times when "Nature and Nature's laws lay hid in night," the story has a happy ending: "God said, 'Let Newton be!' and all was light."[8]

This has not happened, and many important phenomena in special sciences are still unexplainable and unpredictable by the more fundamental theories (Mitchell 2009). Emergence in special sciences has proved to be way more resilient than what early weak emergentists suggested. This resilience indeed begs an explanation. As Fodor (1997, 160–61) nicely puts it: "Damn near everything we know about the

[6] Bedau calls his theory *weak emergence*. I, however, call it *ontological* weak emergence to differentiate it from other accounts of weak emergence.

[7] There are inconsistencies in the field about what is called "weak" or "strong" emergence. I define the terms in the text to avoid any confusion.

[8] Epitaph by Alexander Pope.

https://doi.org/10.1017/psa.2022.5 Published online by Cambridge University Press

world suggests that unimaginably complicated to-ings and fro-ings of bits and pieces at the extreme microlevel manage somehow to converge on stable macro-level properties . . . . On the other hand, the 'somehow' really is entirely mysterious . . . . So, then, why is there anything except physics? . . . Well, I admit that I don't know why. I don't even know how to think about why."

This resilience is a good motivation to ask if there is more to emergence than merely the transient surprise of the ignorant. In fact, betraying Hempel and Oppenheim's expectation, not only did the advances in science not demystify emergent phenomena but it was those very advances that lead to a stronger ontological version of weak emergence. Studies of complex systems through the twentieth century and through today inspired Darley (1994) and Bedau (1997, 2008) to suggest a new kind of weak emergence, namely ontological weak emergence, that while it does not introduce new ontological types, it blames the unpredictability of emergent phenomena on some ontological characteristics of the system. That ontological character is computational irreducibility.

Ontological weak emergence is a property or a behavior of a system that cannot be explained, derived, or predicted except via simulation (Darley 1994; Bedau 1997, 2008). Bedau defines simulation as derivation of the macro-state of the system by going through the course of interactions in its micro-dynamics (Bedau 1997). The course of interactions can be followed in the system itself, its physical replica, or a computational simulation. The important point is that the only way to derive what comes out of the course of interactions is going through that very course of interactions in one setup or another.

Note that it is not a necessary fact that we need to go through the micro-level interactions to derive the macro-level facts of a system. For example, consider the oscillation of an ideal pendulum. To derive the position of the pendulum at time t ($x(t)$), we do not need to crunch through all the forces acting on the bob in every swing until t. We can simply plug the value for t in the following formula and get the position of the pendulum $x(t)$ (A is amplitude, and ω is angular frequency).

$$x(t) = A\cos(\omega t)$$

But such shortcuts are not available for the ontologically weakly emergent, and the only way to derive them is to go through all the forces step by step. Compare predicting the position of a ball in a pinball machine with predicting the position of a bob of a pendulum. Apparently, we know everything about the mechanics of pinball machines. But unlike the case of the pendulum, we cannot come up with a simple formula to tell the position of the ball at t. The only way to know the position of the ball is to play or simulate each round of pinball. Therefore, the position of a ball in a pinball machine at time t is ontologically weakly emergent.

But why is it the case that simulation is the only way to predict the position of the ball in a pinball machine? Arguably, before the invention of trigonometry by Hipparchus (180–125 BC), the shortcut formula for pendulums was not available and simulation was the only way to derive the position of the bob. So, what prevents us from hoping that some future genius will come up with a formula for predicting the position of the ball in a pinball machine, even if it takes devising a whole new branch of mathematics?

https://doi.org/10.1017/psa.2022.5 Published online by Cambridge University Press

In response to this question, Bedau takes recourse to computational irreducibility. He suggests that a phenomenon is ontologically weakly emergent if the path leading to that phenomenon is computationally irreducible. And computational irreducibility guarantees that the only way to predict the phenomenon is going through the original path itself. No shortcuts are possible. In fact, there is such a tight connection between computational irreducibility and ontological weak emergence that Bedau sometimes seems to take the two concepts to be identical (Bedau 2002, 18): "The behavior of [ontologically] weakly emergent systems cannot be determined by any computation that is essentially simpler than the intrinsic natural process by which the system's behavior is generated. Wolfram ... terms these systems 'computationally irreducible.'"

Computational irreducibility is an ontological character of the system and therefore, an emergence arising from computational irreducibility is tied to the ontology of the system. Thus, it is not merely in the eyes of some beholder and is here to remain in face of any future advances. Computational irreducibility guarantees that not just us, the mortal humans of the twenty-first century, but no pinball genius in the future, and no pinball Laplacian demon, can tell the position of the ball without simulating it. Forever and everywhere, simulation is the only option. Or so the supporters of ontological weak emergence claim.

At the core of Bedau's account sits the assumption that computational irreducibility means guaranteed observer-independent unpredictability. The discussions of the previous sections, however, show that computational irreducibility cannot guarantee unpredictability. Computational irreducibility guarantees impossibility of prediction only with maximal standards, and we saw that impossibility of prediction with maximal standards is not equivalent to unpredictability. Computational irreducibility may be able to demonstrate that one cannot predict the position of the ball in a pinball machine under all scenarios, after infinite time, and with infinite accuracy, precision, and certainty, unless one simulates the play. But computational irreducibility has nothing to say against the possibility of a pinball genius, let alone a pinball demon, predicting the position of the ball with strikingly high, yet sub-maximal standards. Computational irreducibility is consistent with the position of the ball being predictable for all that matters.

Avoiding this objection may be the motivation that makes Bedau emphasize *accuracy* and *completeness* in his relatively more recent formulation of ontological weak emergence, saying that it is the accurate and complete derivation of an ontologically weakly emergent phenomenon that is impossible except by simulation (Bedau 2008). But the emphasis on accurate and complete derivation addresses the objection only by trivializing the concept of unpredictability, and consequently the concept of emergence. There is not even a single case of real-world prediction in which one predicts the state of a system with perfect completeness and accuracy. It trivially holds that the only way to know all future states of any physical system with one hundred percent accuracy and completeness is to run the system itself. But by defining emergence in this way, any physical system whatsoever would turn out to be ontologically weakly emergent. Such an all-inclusive theory of emergence tells hardly anything interesting about the emergent phenomena.

Huneman adopts Bedau's idea but goes one step further. He suggests that on top of unpredictability except by simulation, the nontrivial cases of ontological weak

https://doi.org/10.1017/psa.2022.5 Published online by Cambridge University Press

emergence should satisfy one additional criterion. They should show stable higher-level regularities. He again refers to computational irreducibility to explain how an unpredictable micro-level results in a predictable macro-level regularity. He claims that the micro-level is computationally irreducible and hence, unpredictable, but the computational irreducibility is lost once we go to the higher-level (Huneman 2008, 2012). For example, we saw that coarse-graining transforms a computationally irreducible system to a computationally reducible one. Through a mechanism like coarse-graining, so the explanation goes, we transit from the computationally irreducible micro-level to the computationally reducible and, hence, predictable macro-level.

Similar to Bedau's, a key assumption in Huneman's account is that computational irreducibility means guaranteed observer-independent unpredictability. It is computational irreducibility that supposedly guarantees that emergent phenomena are unpredictable at the micro-level. But we saw that this assumption does not hold, and computational irreducibility cannot guarantee Huneman's criteria of unpredictability on the lower level. His account faces the same problems as Bedau's on that level.

An alternative interpretation of Huneman's account in which the emphasis shifts from predictability on the lower or the higher levels to predictability with maximal or sub-maximal standards seems to fare better. On this interpretation, emergence is unpredictability with maximal standards because of computational irreducibility, while being predictable with sub-maximal standards via processes such as coarse graining that break computational irreducibility. Note that the only role that computational irreducibility plays in this version of Huneman's account is guaranteeing unpredictability with maximal standards, and it is a role that it indeed can play. This guaranteed unpredictability with maximal standards, however, is useless in delineating emergent phenomena in practice, because any natural phenomena you name is practically unpredictable with maximal standards.

This interpretation can at most show a purely theoretical difference between the emergent and the non-emergent, and allows the two classes of phenomena to look exactly the same in the real world. The view, therefore, cannot explain any observable difference between the emergent and the non-emergent phenomena. And the other way round, the practical and observable differences between the emergent and the non-emergent phenomena cannot provide any evidence for this purely theoretical view. The emergence debate, however, is motivated by the supposedly observable and practical differences between the emergent and the non-emergent. With regards to these most eye-catching puzzles of emergence, this purely theoretical approach seems irrelevant and inadequate.

On a third interpretation of Huneman's account, we can focus on predictability after coarse graining rather than unpredictability before it. On this interpretation, we acknowledge that all natural phenomena are unpredictable with maximal standards, and we delineate the emergent as those special ones that can be made predictable by coarse graining. The immediate challenge would be to set a cut off for the amount of coarse graining allowed before an emergent phenomenon becomes predictable. To avoid this difficult challenge, we can define emergence as a graded concept, suggesting that different phenomena show different amounts of emergence depending on the amount of coarse graining needed before they become predictable. It is an intuitively appealing idea, and I think it is a step in the correct direction as it

https://doi.org/10.1017/psa.2022.5 Published online by Cambridge University Press

introduces a graded emergence that reflects the graded nature of predictability. The problem, however, is that it is very hard, maybe impossible, to formalize it as a form of computational emergence. In the next section, I entertain the idea of graded emergence and show that the theory behind computational emergence is too stiff to accompany the moves toward graded emergence.

5. Degrees of emergence

One way to deal with the problems arising from the graded nature of predictability is to describe emergence as a graded concept. The promoters of ontological weak emergence have already proposed this idea, though mostly passingly. For example, in his more recent works, Bedau (2008) suggests that ontological weak emergence comes in degrees, and that phenomena can be more or less emergent. This is a move in the correct direction as it mirrors the graded nature of prediction. The problem, however, is that computational irreducibility cannot accompany emergence in this move. Computational irreducibility works only in extreme and maximal conditions, and it is silent about what falls in the middle. By claiming a graded emergence, Bedau loses the ontological support of computational irreducibility for his theory. This is a heavy loss. After losing the ontological support of computational irreducibility, it is not clear what the advantage of computational emergence would be over the other humbler non-ontological versions of weak emergence. In fact, other more subjective accounts of weak emergence are probably in a better position to define a graded emergence based on grades of predictability, because the spectrum of prediction depends not only on the characteristics of the object of prediction, but also on the subject's epistemic goals and capacities.

The graded version of Huneman's account faces similar problems. The degrees of emergence in Huneman's account start after coarse graining where we can no longer rely on computational irreducibility. We need to devise other computational concepts to formalize and quantify the degree to which a phenomenon becomes predictable after coarse graining. Without such formalization and quantification, the graded version of Huneman's account would not have any ontological and computational element and, hence, would not have any tangible advantage over non-ontological types of weak emergence. The non-ontological types of weak emergence might be even better suited to describe the degrees to which a phenomenon shows emergence after coarse graining because they can more easily accommodate the subjective aspects of predictability.

Is there a way to formalize and quantify degrees of emergence in Bedau's or Huneman's accounts? Hovda (2008) has attempted to build such formalization and quantification. His attempt, however, more than anything shows the seemingly insurmountable challenges facing such formalization and quantification. Hovda adopts Bedau's notion of emergence as derivability only via simulation (s-derivability) and suggests the degree to which a phenomenon is emergent corresponds to the amount of simulation, or more precisely, the amount of computation needed to derive that phenomenon. The more computation is needed, the more is the phenomenon emergent.

Despite its apparent simplicity, there are serious problems when it comes to the details of Hovda's formalism. The most important problem that Hovda himself

https://doi.org/10.1017/psa.2022.5 Published online by Cambridge University Press

explicitly acknowledges is that the amount of computation needed to derive a phenomenon depends not just on the nature of that phenomenon, but also on somewhat arbitrary choices, such as the formal language used to describe the system and the way one defines and counts the steps of the simulation. Listing various such difficulties, Hovda (2008, 470) concludes that "we must acknowledge that s-derivability, and amount of simulation required, might be relative to a derivation system." But this means admitting that the amount of emergence a phenomenon shows at least in part depends on our somewhat arbitrary and subjective choices on the derivation system. The amount of emergence would be a side effect of some arbitrary formalism rather than an ontological fact about the phenomenon.

The dependence of Hovda's quantity of emergence on the choice of formalism diminishes not just its theoretical value in describing the nature of emergence, but also its practical usefulness as a measure of emergence. Science uses different formal frameworks to describe different natural phenomena, and therefore, it would not be possible to use Hovda's quantity to compare the amount of emergence across different natural phenomena.

The lessons of Hovda's example are not peculiar to his particular formalism. His work shows how arbitrary and subjective factors bedevil any quantification of weak emergence. Considering these arbitrary and subjective factors, it seems impossibly challenging to quantify emergence as an ontological computational character of the emergent phenomena.

6. Conclusions

Our exploration of various interpretations of computational irreducibility showed that it can guarantee impossibility of prediction only for all cases, or in infinite time, or with infinite precision, accuracy, and certainty. Prediction in its best scientific sense, however, does not happen in these maximal contexts. Every predicting model scores better or worse in terms of the variety of scenarios it can predict, the stretch of time it can look into, and the accuracy, precision, and certainty of its predictions. Even the archetypes of scientific predictability do not score perfectly on any of these dimensions. Yet we deem those scientific predictions *successful* and, accordingly, deem the object of their predictions *predictable*. Therefore, we cannot conclude unpredictability of a phenomenon from a theory like computational irreducibility that shows impossibility of prediction only with maximal standards.

This has important consequences for the philosophical positions such as ontological weak emergence that rely on computational irreducibility to guarantee unpredictability. Ontological weak emergence defines emergence as unpredictability except via simulation. It is computational irreducibility that is supposed to guarantee this unpredictability. Allegedly, computational irreducibility provides an objective and ontological anchor for the unpredictability of the emergent and, thus, frees weak emergence from being merely in the eyes of some observer. With its mathematical power, computational irreducibility acts as the seal of Solomon forcing even the Laplacian demons to kneel before the unpredictability of the emergent. Even the demons cannot predict the outcome unless they simulate.

https://doi.org/10.1017/psa.2022.5 Published online by Cambridge University Press

But the discussions above show that until we forge a stronger formulation, the seal will remain lost in the sea.[9] Computational irreducibility leaves open the possibility that not only the Laplacian demons, but even the humble human observers predict the result of a computationally irreducible process with sub-maximal, yet acceptable standards. Computational irreducibility only shows that we cannot ask for the moon. But we do not need the moon after all.

Acknowledgements. I would like to thank Jacob Stegenga, Tim Lewens, Mohammad Reza Mousavi, Hamed Ghasemieh, Ali Haghi, Adrian Erasmus, and anonymous referees for helpful discussions and/or feedback on earlier drafts.

References

Aaronson, Scott. 2002. "Book Review on A New Kind of Science." *Quantum Information and Computing* 2 (5):410–23.

Bedau, Mark A. 1997. "Weak Emergence." *Noûs* 31 (s11):375–99.

Bedau, Mark A. 2002. "Downward Causation and the Autonomy of Weak Emergence." *Principia* 6 (1):5–50.

Bedau, Mark A. 2008. "Is Weak Emergence Just in the Mind?" *Minds and Machines* 18 (4):443–59.

Berlekamp, Elwyn R., John H. Conway, and Richard K Guy. 1982. *Winning Ways for Your Mathematical Plays Vol 2*. London: Academic Press.

Berto, Francesco, and Jacopo Tagliabue. 2017. "Cellular Automata." In *The Stanford Encyclopedia of Philosophy*, edited by Edward N. Zalta. Stanford: Stanford University Press.

Brier, Glenn. 1950. "Verificationn of Forcasts Expressed in Terms of Probability." *Monthly Weather Review* 78 (1):1–3.

Buss, Sam, Christos H. Papadimitriou, and N. J. Tsitsiklis. 1990. "On the Predictability of Coupled Automata: An Allegory about Chaos." In *Proceedings [1990] 31st Annual Symposium on Foundations of Computer Science*, 788–93.

Chalmers, David J. 1996. *The Conscious Mind: In Search of a Fundamental Theory*. New York: Oxford University Press.

Chalmers, David J. 2008. "Strong and Weak Emergence." In *The Re-Emergence of Emergence: The Emergentist Hypothesis from Science to Religion*, edited by Philip Clayton and Paul Davies, 245–54. New York: Oxford University Press.

Charbonneau, Paul. 2017. *Natural Complexity A Modelling Handbook*. Princeton, NJ: Princeton University Press.

Darley, Vince. 1994. "Emergent Phenomena and Complexity." In *Artificial Life IV, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems,* edited by Rodney Brooks and Pattie Maes, 411–16. Cambridge: MIT Press. .

Davies, Paul C. W. 2004. "Emergent Biological Principles and the Computational Properties of the Universe: Explaining It or Explaining It Away." *Complexity* 10 (2):11–15.

Fodor, Jerry. 1997. "Special Sciences: Still Autonomous after All These Years." *Philosophical Perspectives* 11:149–63.

Habib, Michel, and Juraj Stacho. 2013. "Unique Perfect Phylogeny Is Intractable." *Theoretical Computer Science* 476:47–66.

Hempel, Carl G., and Paul Oppenheim. 1948. "Studies in the Logic of Explanation." *Philosophy of Science* 15 (2):135–75.

Hovda, Paul. 2008. "Quantifying Weak Emergence." *Minds and Machines* 18 (4):461–73.

Humphreys, Paul. 2016. "Inferential Emergence." In *Emergence: A Philosophical Account*, 144–79. New York: Oxford University Press.

Huneman, Philippe. 2008. "Emergence Made Ontological? Computational versus Combinatorial Approaches." *Philosophy of Science* 75 (5):595–607.

[9] The story goes that Solomon possessed a seal ring by means of which he had dominion over all demons. A demon once stole the ring and threw it into the sea. Solomon was thus deprived of his dominion until he found the ring inside a fish (Jacobs and Seligsohn 1906).

https://doi.org/10.1017/psa.2022.5 Published online by Cambridge University Press

Huneman, Philippe. 2012. "Determinism, Predictability and Open-Ended Evolution: Lessons from Computational Emergence." *Synthese* 185 (2):195–214.
Ilachinski, Andrew. 2001. *Cellular Automata: A Discrete Universe*. Singapore: World Scientific Publishing Company.
Israeli, Navot, and Nigel Goldenfeld. 2006. "Coarse-Graining of Cellular Automata, Emergence, and the Predictability of Complex Systems." *Physical Review E* 73 (2):26203.
Jacobs, Joseph, and M. Seligsohn. 1906. "Solomon, Seal Of." In *The Jewish Encyclopedia*, Vol. 11, edited by Cyrus Adler, Leon Hühner, and David Salzberg, 448.
Lewis, Rhyd, and Fiona Carroll. 2016. "Creating Seating Plans: A Practical Application." *Journal of the Operational Research Society* 67 (11):1353–62.
Mitchell, Melanie. 2009. *Complexity: A Guided Tour*. New York: Oxford University Press.
Rendell, Paul. 2011. "A Universal Turing Machine in Conway's Game of Life." In *2011 International Conference on High Performance Computing & Simulation*, 764–72.
Rucker, Rudy. 2003. "A New Kind of Science." *The American Mathematical Monthly* 110 (9):851-61.
Tetlock, Philip E, and Dan Gardner. 2016. *Superforecasting: The Art and Science of Prediction*. London: Random House.
Turing, Alan Mathison. 1937. "On Computable Numbers, with an Application to the Entscheidungsproblem." *Proceedings of the London Mathematical Society* 2 (1):230–65.
Wolfram, Stephen. 1984. "Cellular Automata as Models of Complexity." *Nature* 311 (5985):419.
Wolfram, Stephen. 2002. *A New Kind of Science*. Champaign, IL: Wolfram Media.
Yanofsky, Noson S. 2016. *The Outer Limits of Reason: What Science, Mathematics, and Logic Cannot Tell Us*. Cambridge, MA: MIT Press.
Zhu, Liping, Song-Ju Kim, Masahiko Hara, and Masashi Aono. 2018. "Remarkable Problem-Solving Ability of Unicellular Amoeboid Organism and Its Mechanism." *Royal Society Open Science* 5 (12):180396.
Zwirn, Hervé. 2013. "Computational Irreducibility and Computational Analogy." Preprint, submitted April 18, 2013. https://arxiv.org/abs/1304.5247.
Zwirn, Hervé, and Jean-Paul Delahaye. 2013. "Unpredictability and Computational Irreducibility." In *Irreducibility and Computational Equivalence. Emergence, Complexity and Computation*, Vol. 2, edited by Hector Zenil, 273–95. Berlin: Springer.

Cite this article: Tabatabaei Ghomi, Hamed. 2022. "Setting the Demons Loose: Computational Irreducibility Does Not Guarantee Unpredictability or Emergence." *Philosophy of Science* 89 (4):761–783. https://doi.org/10.1017/psa.2022.5