Empirical equivalence and distinguishability: a computational complexity approach.

August 30, 2021

Contents

1 Introduction **1**

2 Description of the proposed numerical simulations **4**
- 2.1 General idea 4
- 2.2 Type of models. 4
- 2.3 Computation and robots 5
- 2.4 Time reversibility of the evolution rule 6
- 2.5 An example of models that are expected to be equivalent 6

3 Dispute about the equivalence of two models **7**
- 3.1 A new perspective on empirical equivalence 7
- 3.2 Embedding a Turing machine in the model: encoding and decoding 8
- 3.3 Embedding a robot in the model 9
- 3.4 Object of the dispute 9
- 3.5 Difficulties in settling a dispute 10
- 3.6 Method based on computational complexity 12
- 3.7 Procedure for settling a dispute 13

4 Transformations: do they ensure equivalence? **13**
- 4.1 Definitions 13
- 4.2 Polynomial-time invertible transformations ensure equivalence . . 14
- 4.3 A (complex enough) transformation connects couples of models . 15

5 Conclusions **16**

1 Introduction

Scientists check their theories by comparing the predictions with observations and experimental results. This operation is routinely performed by scientists, but subtleties arise when it is discussed in depth. The philosophical background of this discussion relies on the concept of “data equivalence”.

Duhem has pointd out that, for any theory T, a data equivalent rival theory T' can be found. The same author also discussed the difficulties in the comparison between a theory and experiments: the experiments often rely on additional hypothesis, making a direct comparison problematic [1]. An extremely critical point of view is expressed by the Quine-Duhem thesis, radically questioning the possibilty of comparing an experiment with a single theory or with a small set of theories. Such difficulties lead to the development of the deeper concept of "empirical equivalence" [2, 3, 4, 5].

Data equivalence and empirical equivalence have been discussed in particular in relation with realism [6, 7]; shortly, it is discussed if a realistic point of view can be kept also when there are multiple rival theories which are either data equivalent or emipirically equivalent.

Roughly speaking, the difficulties in comparing experiments to theories arise due to the intricate relations among the theories describing the various aspects of the experiment and the hypothesis on the behavior of the instruments; the situation becomes even more complex when possible non-deterministic phenomena are taken into account, including experimental errors, quantum non-determinism and non-determinism of statistical theories such as statistical mechanics and thermodynamics.

In this paper, I propose to tackle the problem from an alternative point of view. I suggest to hypothetically consider a world that is determined by known rules, and to discuss what its inhabitants would be able to discover about the rules. Although this situation is different from real science (we will start from a privileged situation, in which we already know the "real" theory), I argue that this study could give hints about significant real problems of epistemology.

This approach is inspired by the widespread use of computer-based simulations in every field of science, from fundamental physics (e.g. numerical calculations of fundamental particle interactions) to biology (e.g. flock dynamics). The target of the simulation is not necessarily limited to an accurate modelling of a real system, but is sometimes used to get a general understanding of a concept. An example is the theory of the so-called "sand piles" [8]: although the similarity to real piles of sand is limited, they have been used as a prototype for the concept of self-organized criticality, which is common in nature.

The usual application of computer-based simulation in science is to calculate results of a model and compare them with experiments. This application does not significantly involve epistemological problems: it is assumed that the scientist performing the experiments and the simulations is "external" with respect to both. I underline that the target of the present study is different, i.e. addressing the issues of epistemology by simulating not only the system, but also the process of making experiments and drawing conclusions. More imaginatively, the aim is to try to understand what the inhabitants of a hypothetical world could understand about the physical laws of their world, by simulating not only the phenomena but also the inhabitants themselves. This can be described as inspecting the "internal" point of view. Despite the widespread use of computer simulations, to my knowledge this approach has never been attempted in the field of philosophy of science, at least in the form that I am proposing here.

The idea of simulating a world with a computer, and studying the limits to the knowledge of its inhabitants, could appear unfeasible in practice: the simulation of a whole experimental setup, and, in principle, of the scientist performing the experiment, including his brain, is beyond the capacity of computers. I will limit the discussion to very simplified simulations and entities performing experiments (robots rather than humans). The very limited target is only a first attempt in this direction, which can be expanded in the future. Although limited, the discussion already captures some of the features of real systems. Moreover, and more significantly, I will not present results of computer simulations, but rather I will focus on the definition of the problem and on the derivation of some general laws. In other words, I will present a discussion of "thought computer simulations" (where the term "thought" has a meaning similar to the meaning it has in the expression "thought experiments"). This way of proceeding has been already proved fruitfully, for example in the case of the theorems obtained on the Turing machines: they were actually "thought experiments" discussed in advance with respect to the real development of computers; they gave interesting results independently of the practical realisation of the machines.

In Sect. 2, I describe the proposed thought computer simulations. The simulation includes the hypothetical physical world and an entity in it, which can be thought about as a robot, which performs experiments. The aim of the experiments performed by the robot is to discover the governing rules of the simulation; more precisely, I will focus on the possibility of distinguishing between two models, i.e. to understand if the two models are empitically equivalent or not. In Sect. 3 I show that drawing a conclusion on the empitical equivalence of two models is tricky and some subtleties arise; I show how a proper discussion of the equivalence can be performed and how to settle a dispute on the topic. This discussion is based on computational complexity theory [9]. It has been already noticed that this theory provides a deep characterisation of physical concepts [10].

In physics, an important role is played by "transformations" which connect different models. They are often assumed to imply equivalence (in some rather vague sense). For example, the change of reference system is actually a transformation between two systems which are mathematically different, but are physically the same. A similar role is played by canonical transformations in Hamiltonian mechanics. Section 4 discusses what is implied by the presence of a transformation between two models. In particular, I mathematically prove a sufficient condition for empirical equivalence of two models, valid for the thought computer simulations under discussion, based on the presence of a transformation between the models with a property expressed in terms of computational complexity. This is a first result showing the power of the proposed approach.

I emphasize that the approach based on the simulation of the whole system, including the experimental apparatus, is relevant not only for reaching a deeper understanding of epistemology *per se*, but also for the direct consequences on science. Indeed, there are situations in which the relation between the object of the experiment and the experimental system, possibly including the scientist performing the experiment, has not been clarified. One example is the quantum mechanics: the various interpretations and the various alternative theories proliferated because of the apparent effect of the measuring apparatus on the outcome of the experiment. Another example is the concept of entropy: it has been described as a measure of non-knowledge, thus it can be discussed in the framework of the proposed study.

2 Description of the proposed numerical simulations

2.1 General idea

Here I call a "model" a mathematical model characterized by a state $s(t)$ which changes with time t. The model is defined by a law which fully determines $s(t)$ once the initial state $s(0)$ is given. Under this definition we find, for example, the classical mechanics; the wavefunctions of quantum mechanics can be also described in this way, although the outcomes of a measurement do not fit in this view.

We see that the model fully defines the world that we aim at studing, once the initial state is given. The proposed computer simulations consist in calculating $s(t)$ from a given initial $s(t = 0)$. In order to investigate the operation of performing experiments, I propose to embed a "robot" inside the model, i.e. to give a particular initial state which describes something that can be interpreted as a machine able to perform mechanical operations. The machine must be able to execute a program and perform calculations, in order to draw conclusions.

It is important to notice that the models do not need to give results similar to real experiments. The idea, indeed, is to study hypothetical worlds with rules which can be different from each other, and from the real world, with the aim of getting a better understanding of the process of making experiments.

2.2 Type of models.

Fundamental physical models are based on a continuous description of space and time. Here, I focus instead on discrete models, in which the state s is represented by a set of bits (possibly, infinite) which change in discrete time steps. This simplification allows us to apply more easily the concepts of computational complexity in the following.

The mathematical notation used here is the following. The state s is defined by the bits s_j. The time is discrete. The rule governing the model is defined by an "evolution rule" (a function) F:

$$s(t+1) = F\left[s(t)\right] \tag{1}$$

As an example of such models, I mention the cellular automata. Such models are based on a set of cells, distributed on a lattice with defined neighborehood relations, in turn, defining the dimensionality. Each cell, at a given time, has a cell state, expressed by a number. The representation of the cell states of all the cells as a sequence of bits is the state s of the cellular automaton. At each time-step, the new state of cell j, $s_j(t+1)$, is calculated based on the current states (at time t) of the cell j itself and of the neighboring cells. A famous example of cellular automaton is the game of life (GoL) [11]; various cellular automata are reviewd in Ref. [12]

Another example of discrete model is the sequence of snapshots of a computer. The state s represents the content of the memory and of the CPU registers, while the evolution rule is defined by the machine language of the CPU. More formally, we can think of the snapshots of a Turing machine: s codes the contents of the tapes and the Turing machine state. In this case, the evolution rule is determined by the Turing machine itself; the above-mentioned example of the computer is similar to the case of the universal Turing machine.

2.3 Computation and robots

To our purpose, the model must be powerful enough to enable the description of the experimental apparatus and of its control logic. The cellular automata are actually quite powerful. Indeed it is known that many of them are "Turing-universal": roughly, it means that it is possible to implement any arbitrary Turing machine in it, so that the automaton itself behaves as a universal Turing machine. In other words, it is possible to give a suitable initial lattice which evolves similarly to a Turing machine (or a real computer) [13]. Examples of Turing-universal cellular automata are the above-mentioned GoL [14] and "Rule 110" [15].

The other example of discrete model given in Sect. 2.2, the snapshots of a computer or of a Turing machine, is also powerful: indeed, a universal Turing machine is used, by definition, to define the Turing-universalily.

With a highly imaginative representation, the idea of "embedding a robot in the model" would be to chose the initial state so that a machine, similar to a robot, is simulated inside the model. The control of the robot is through a computer, embedded in the model too, which is able to execute a program, having the power of a universal Turing machine. In addition to be able to execute the instructions needed by the requirement of Turing-universality, the machine will be able to execute some special instructions, corresponding to the operation of the robot: they will control the actuators and receive the feedbacks from sensors.

More concretely, the idea suggested here is that of a universal Turing machine, implemented inside the model, having a (finite) set of additional function which represent the interaction of the Turing machine with the rest of the model, the enviroment in which the Turing machine is embedded. The universal Turing machine plus the special interaction functions will be called "the robot", althought such functions can be thought of as relatively simple and physically very different from a robot. The researcher (the real human who wants to investigate the procedure of making an experiments by using the methods discussed here) chooses a model, constructs the initial state implementing the universal

Turing machine, uploads a program for the robot, runs it and looks at the result of the calculation of the machine (or, better, of the operation of the robot). Clearly, by simply interacting with a pure universal Turing machine, the researcher would never learn anything on the evolution rule of the model: the special interaction functions must be used by the program.

This approach has never been proposed previously, nor explicit examples have been given, so it is not even clear if such a robot can be simulated in a model. However, situations bearing some similarity have been considered. For example, in special cellular automata, it was possible to simulate a self-replicating robot [16]. On the other hand, the possibility that such a robot can be embedded in the model, and that it can actually discover some of the rules of the model itself, is the topic of the very preliminary discussion presented here.

2.4 Time reversibility of the evolution rule

Mamy fundamental theories are time-symmetric, e.g. Newtonian mechanics, i.e. the time-evolution equations remain the same if time t is changed with $-t$. Theories derived from a time-symmetric theory, by averaging or coarse-graining, can be non-time-symmetric, such as hydrodyamics or statistical mechanics. Here I will assume a weaker condition, called "reversibility": it means that it is possible to evaluate the evolution of a state $s(t)$, at time t, for instants both following and preceding t.

In discrete systems, considered here, reversibility is expressed by the condition that F is invertible, i.e. the inverse of F exists, F^{-1}, so that it is possible to write:

$$s(t-1) = F^{-1}\left[s(t)\right] \tag{2}$$

Reversible Turing machines (and, more in general, reversible computers) have been studied [17], although (regular) Turing machines are not reversible. The phenomena on which computers are based are thought to be reversible. In real computers, each computer state (Turing-state) corresponds to a moltitude of states of the physical system (micro-states); this explains how the time-reversible physical substrate can generate a non-reversible computer.

In analogy with the real computers, I will assume that the discrete models considered here are time-reversible, that the Turing machines implemented in them are non-reversible, and that each Turing-state corresponds to a moltitude of micro-states. The reason for this choice will become more clear in the following, in relation with the problem of simulation of a model in another.

2.5 An example of models that are expected to be equivalent

At this stage of description of the proposed approach, it has not been explained yet if it is possible to find two models that can be proved to be equivalent, nor if two non-equivalent models can be found. To clarify this point, I show that there are couples of models which should result equivalent, if the approach correctly represents the equivalence from the internal point of view. An example is the following.

Example 1. Given an evolution rule F, a different evolution rule $\bar{F}$ is obtained by "flipping all the bits":

$$\bar{F}_j(s_i) = \neg F_j(\neg s_i) \tag{3}$$

where the operator $\neg$ represents the negation of a bit.

It can be easily seen that, if $s_j(t)$ is a sequence of states obeying the evolution rule F, then $\neg s_j(t)$ obeys the bit-flipped evolution rule $\bar{F}$. In other words, the difference between the two models is simply the representation of the bits, which appears only from an external point of view: intuitively, we expect that any F and $\bar{F}$ will result equivalent.

3 Dispute about the equivalence of two models

3.1 A new perspective on empirical equivalence

Usually, the numerical simulation is used to calculate results of a model; the results expected from the model are then compared with the results of real experiments. The researcher has an *external point of view*, i.e. it sees from outside both the experimental system and the numerical simulation. If two simulations of models α and β give the same experimentally testable results, then the models are judged empirically equivalent.

The present approach is radically different. I imagine that the two models α and β are simulated by the computer, including the experimental apparatus and a robot making the experiment, and I wonder if it is possible to perform an experiment which gives different results in α and β. If not, the two cellular automata are deemed empirically equivalent. This represents the new perspective on empirical equivalence, based on an *internal point of view.* In the following, the term "equivalence" will refer to this notion of equivalence, evaluated by simulating the robot performing experiments, and will thus refer to the "internal point of view".

It is important to emphasize here that I do not expect that the robot itself performs a heuristic process to discover the evolution rule F: this would require an investigation of artificial intelligence. Instead, the interpretation of the results of the experiments is left to the researchers, the (real, human) scientists that are making the computer simulation.

In order to clarify the concept, I visualize it as a dispute between two researchers, Φ and Ψ, about the equivalence of two models α and β: Φ thinks that α and β are equivalent, Ψ thinks that they are non-equivalent. Here, the equivalence has the above-described meaning of "equivalence from the internal point of view". Simply inspecting some sequences of states $s(t)$ was not enough to solve the dispute, so they try to inspect the "internal point of view" by embedding a robot in the models α and β and controlling it through a program. The two researchers write various programs for the robot and observe the results of the machine calculation and robot operation. By this interaction, the researchers try to prove their opinion. This section is devoted to the formalization of such a procedure and to the discussion of some problems which arise in the analysis of the results.

3.2 Embedding a Turing machine in the model: encoding and decoding

The robot embedded in the model, having the task of performing experiments, must have a control logic. I suggest that this control logic must have at least the power of the Turing machine; the reason will be clear in the following. If it is possible to describe any arbitrary Turing machine in the model, as we aim to do, then the model is said to be Turing-universal.

The concept of "universal machine" was developed by Turing, who showed the existence of a machine (nowadays called "universal Turing machine") that can be programmed in order to behave as any other desired Turing machine [18]. Roughly, the machine takes as inputs two strings, the first representing the program p to be executed and the second the data d on which the program must run. In the case of the machine proposed by Turing, it was clear from the definition that his "universal machine" was able to simulate the behaviour of any specific Turing machine.

When we want to discuss the Turing-universality of a given cellular automaton, however, subtleties arise [19]. The researcher who wants to prove that a cellular automaton is Turing-universal must provide two functions, the encoder E and the decoded D. The former translates the couple p, d into an initial state $s(t = 0) = E(p, d)$; the latter takes a state $s(t)$ at a given time t as argument and returns the halting state of the machine, i.e. $D[s(t)]$ can be either "accept", "reject", or "not finished".

The above-mentioned subtleties stem from the possibility that the researcher, who provides E and D, could conceal the execution of the program p inside E or D. For example, E could execute the program p on the data d, calculate the outcome, and return it encoded into $s(t = 0)$; then, D simply decodes $s(t = 0)$, thus giving the corret outcome of the calculation; however, in this example it is clear that no calculation was performed by the cellular automaton itself.

In order to overcome this proble, the definition of "Turing universality" require that E and D have additional properties. Rougly, it is requested that E and D have a complexity smaller than the Turing machine, so that they cannot perform an arbitrary program p on data d.

- In Turing-universality, the coding and decoding functions are required to be (total) computable functions [20, 21] (roughly, they can be computed in a given finite time for every input). The implemented Turing machine instead calculates partial functions (roughly, functions for which it is not guaranteed that the calculation halts)
- In related case of circuit-universality [22], the coding and decoding algorithms are required to be in the complexity class **NC** (see Ref. [9]).

> Instead, the calculation of the output $c(i)$ of the implemented circuit is a **P-complete** problem. It is conjectured that **P-complete** problems are more complex than **NC**, thus E and D cannot conceal the operation of calculating the output of the program.

These requirements prevent to conceal the calculation performed by the implemented Turing machine inside the encoding and decoding operations: we can say that the calculation of the output of the Turing machine or of the circuit is genuinely performed by the cellular automaton.

For the sake of the following discussion, I will assume that E and D can be calculated in polynomial time in the size of their inputs. In this way, E and D will not be able to conceal the calculation of, e.g., an EXP problem, which can however be calculated by a computer. Here, it is worth remarking that it has been proved that there are problems that are in EXP but not in P.

It must be remarked that the inspection of the code for the calculation of E and D could not naïvely lead to a clear answer to the question whether it conceals the execution of the program p; it is even possible that the researcher her/himself who provided E and D is not aware that they actually conceal the calculation of p. This is the reason why the above-mentioned procedure, based on the calculation complexity, is needed.

3.3 Embedding a robot in the model

Up to now, I only discussed the embedding of a Turing machine. Embedding a robot has been also discussed in literature in some specific cases, such as for the replicating automa [16]. I do not show the feasibility of such operation in a general case. Rather, I assume that, if this is feasible, the control of the robot will take place through the implemented universal Turing machine. In the case of a "normal" universal Turing machine, the program p is the description of a Turing machine, the one that must be calculated. In the case of the robot, the universal Turing machine will have a given set of additional functions, which operate on the robot.

Less formally, we can think of p as a program written, e.g., in C. With the standard set of C commands, we can write a C program which performs the calculation of any Turing machine. But now we add a set of additional functions to operate the robot, interacting with actuators and sensors.

After this stage, the researcher will be able to interact with the model by writing programs p (possibly operating the robot) and looking at the output of the program.

3.4 Object of the dispute

I discuss here what happens when there is a dispute between two researchers, Φ and Ψ, about the equivalence of two models α and β. I assume that Φ thinks that α and β are equivalent, while Ψ thinks that they are non-equivalent. Φ and Ψ propose encoding and decoding procedures, E and D, for the two models α

and β (so we have E_α, E_β, D_α, and D_β). Then they write programs p and run them in the two models. Φ aims at showing that the results are always equal in the two models, instead Ψ aims at showing that at least one program gives different results in the two models. It must be emphasized here once again that the programs are actually "robot programs", since their language admits a set of instructions for operating the robot, and are not just programs for a universal Turing machine.

3.5 Difficulties in settling a dispute

Naïvely, it could be argued that the inspection of the internal point of view could be discussed simply by starting with some initial state, originating a certain phenomenon that is investigated, and by observing the time sequence originating from it. However, it is possible that the two researchers Φ and Ψ apply a misleading reasoning. It must be emphasized that such misleading resonings could be not easy to recognize. To give an idea of what I mean for such difficulties, I could say that they are similar to the ones arising in the case of the rigorous definition of Turing-universality discussed in Sect. 3.2. As in that case, it is possible that apparently innocuous procedures actually conceal some effect that make the whole reasoning completely wrong. It must be emphasized that it is possible that such faulty reasonings are done without the aim of cheating; they cannot be ruled out unless a clear formalization of the requirements is done.

To this aim, I list the possible faulty reasonings and the possible solutions. After this, I will describe a whole procedure aimed at avoiding the faulty reasonings.

1: Direct inspection of the state sequence The researcher Ψ proposes two initial states, for α and β, which, in the idea of the researcher, represent similar experiments run in different models. Then, Ψ inspects the two time sequences. They are, of course, different, because the evolution rule is different. This is given by Ψ as a proof that the two models are not equivalent.

In order to see that this reasoning is faulty, we can take two models of Example 1, i.e. a model α and the same with flipped bits as β. Then Ψ proposes a given initial state s for α and $\neg s$ for β. Their time sequences are different, thus Ψ could argue that α and β are not equivalent. However, this reasoning is likely not representing the internal point of view, since Ψ had direct access to the *representation* of the models, which is relevant only from the external point of view.

The idea of embedding the robot in the model was aimed at avoiding such a faulty reasoning. The researchers communicates with it only through the program p and its result, so they do not have direct access to the representation of the model.

However, even with this approach, the discussion could be not easy. The researchers Φ and Ψ could make use of misleading methods to prove their thesis, of which some are quite trivial. It is thus necessary to find a procedure for settling a dispute, which avoids the use of such tricks.

2: Hard wiring The encoding and decoding algorithms E and D are written by one of the researchers Φ and Ψ. They are necessarily different for the two models; also the functions enabling the interaction with the robot are implemented differently in the two models. Thus it is possible that Ψ implements two different robot interaction functions, returning data which directly identify the model, without making any real robot operation.

As an example, Ψ could implement a hard-wired function which returns 0 in the encoder E_α for α and 1 in the encoder E_β for β (with no further operation). This will allow Ψ to claim that the two models are non-equivalent, also in the case of *identical* models.

In order to avoid this faulty reasoning, it is necessary that both Φ and Ψ have the possibility of providing the encoders and the decoders. If Ψ provides two encoders E_α and E_β with differently hard-wired robot-interaction functions, then Φ can show that the reasoning is faulty simply providing an alternative encoder E'_β, in which the robot-interaction functions are instead hard-wired to the same results of E_α.

3: Arbitrary interpretation of the outcome of the calculation One of the researchers could write a decoding algorithm D such that it does not give the real result of the calculation or operation really performed in the model, by analyzing the state $s(t)$ at the end of the simulation, but rather calculates a result (independently of the state of the model s) which induces to believe that the two models are equivalent (or non-equivalent).

This situation is similar to the problem which arises in the definition of Turing-universality: the programs E and D could conceal some kind of calculation of the desired result. In analogy with the solutions proposed for that case, in order to avoid this problem, we require that E and D operate in polynomial time in the size of the program. Then the researchers will check if the result of an EXP-complete problem is still correctly calculated. This point is particularly significant and will be discussed in Sect. 3.6.

4: Simulation of a model inside another one The models considered here can be calculated by a Turing machine in a finite number of steps. On the other hand, both α and β can receive initial data such that the time sequence generated from such initial states contains the information of the time-evolution of a Turing machine (they are Turing-universal). So it is always possible to give an initial state of α such that it represents the calculation of the time sequence of β. In other words, whatever E and D are proposed by Ψ for β, Φ can always propose two E and D for α which give exactly the same result, by embedding a universal Turing machine in α, with a program which calculates the evolution of β.

Here, I do not aim at discussing the idea that our universe is a computer simulation. This discussion is present in literature [23] but it does not have anything to to with the present paper. Rather, the problem of simulation is a possible faulty reasoning which could be applied by Φ in order to try to show that α and β are equivalent.

This problem is avoided thanks to the property that each Turing-state corresponds to several micro-states. If α and β are equivalent, then the number of mocro-states corresponding to each Turing-state are the same. If α is simulating β, the number is larger, since each micro-state of β will be simulated by a Turing-state of α, which, in turn, will correspond to even more micro-states of α.

3.6 Method based on computational complexity

I propose that the point 3 in Sect. 3.5 can be solved with a method based on computational complexity analysis.

The problem addressed here refers to the difficulty of unambiguously associating a state $s(t)$, reached at the end of the simulation, to the actual outcome of the executed program p. This association should be performed by the decoder D; we want to avoid that it genuinely reports the decoded exit status of the program, but it could instead conceal some calculation, enabling D to first discriminate between the model α and β, and then transferring this information as output.

The proposed method to prove that the outcome of the program is genuinely evaluated and reported by D is based on the computational complexity. It aims at certifying the output of a given robot program p ("accept" or "reject") without relying on the inspection of E and D. Imagine that the researcher Ψ wants to prove that the outcome of a given robot program (with finite running time) p is "accept". Ψ chooses an EXP-complete problem and writes a new program, p', which is a modification of p. This new robot program takes the representations of two instances of the chosen EXP-complete problem as input, e_1 and e_2, an operates as follows:

1. p' executes p until the state "accept" or "reject" is reached. As requested, p operates in finite time.

2. If the outcome of p is "accept", then p' proceedes by solving e_1, if the outcome of p is "reject", then p' proceedes by solving e_2. This can take exponential time in the input length.

3. p' returns the outcome of the calculation of e_1 or e_2.

Then, Ψ shows that the output of p' is always the solution of e_1 (for whatever input e_1, e_2). This is a proof that the output of p is genuinely "accept": E and D operate in polynomial time, thus they cannot conceal the calculation of the solution of e_1 or e_2, which are instances of an EXP-complete problem.

It is possible to make a step forward in this direction. It can be argued that E could operate such that both e_1 and e_2 are solved by the model, so that D could be able to cheat, giving the solution of e_1 although the output of p is "reject". If this were the case, the EXP-complete problem would be genuinely solved by the model, and D would simply choose the solution irrespectively of the output of p. It is however easy to check if Ψ is cheating in this way. Indeed the final state $s(t)$ should always contain the solution of both e_1 and e_2, which can be decoded in polynomial time.

3.7 Procedure for settling a dispute

I remind that there is a dispute between Φ and Ψ, about the equivalence of two models α and β. Φ thinks that α and β are equivalent, while Ψ thinks that they are non-equivalent.

Considering the possible faulty reasonings described above, a fair way of settling the dispute should follow this scheme.

1. Ψ proposes two couples of E and D (E_α, E_β, D_α, and D_β), for α and β, and shows a program p which has different outcomes in α and β. (Why does Ψ start? Because it is always easy to build two couples of E and D which do not give differences: it's enough that the robot-interaction functions do not perform any operation, so that they simply implement a Turing machine without any robot)

2. Φ accuses Ψ to use the trick 3 (arbitrary interpretation of the state). Ψ proposes a set of programs p, which run a different EXP-complete problem in α and β, e_1 and e_2, respectively. This calculation cannot be concealed in E or D (because they operate in polynomial time). If Ψ is unable to do so, then the dispute is resolved in favour of Φ, else we go on.

3. Φ objects that maybe E and D actually compute both e_1 and e_2, and D_α reports the result of e_1, while D_β report the result of e_2. Ψ challenges Φ to prove this, by proposing D_α and D_β which report the exchanged results, e_2 and e_1 exchanged. If Φ is able to do so, then the dispute is resolved in favour of Φ, else we go on.

4. Φ accuses Ψ to use the trick 2 (hard wiring) and tries to build E_β and D_β (only for β, not for α) such that they give the same results of E_α and D_α in α. If Φ is not able to do so, then the dispute is resolved in favour of Ψ, else we go on.

5. Ψ accuses Φ to use the trick 4 (simulation). The number of micro-states corresponding to each Turing-state is inspected. If it is the same, then the dispute is resolved in favour of Φ, else in favour of Ψ.

4 Transformations: do they ensure equivalence?

4.1 Definitions

Transformations play an important role in physical theories. The same mathematical model can be represented in different coordinate systems, resulting in different mathematical formulations, which, however, do not change the underlying meaning. A change of coordinates is a transformation, connecting the different mathematical models. It is often implicitly assumed that a change of coordinates transforms a model into an empirically equivalent one. However, the existence of a transformation between two models in general does not ensures that they are empirically equivalent; this has been discussed in particular for quantum theories [24].

In the context of the present discussion, a transformation T is here defined as a function which transforms states of the model into states: $s' = T(s)$. Let us take two models characterized by evolution rules F and F' and consider two time sequences $s(t)$ and $s'(t)$, which obey F and F', respectively. We say that the time sequence $s'(t)$ is the T-transform of $s(t)$ if:

$$s'(t) = T\left[s(t)\right] \tag{4}$$

If this is true for whatever couple of time sequences obeying F and F':

$$F'\left[T(s)\right] = T\left[F(s)\right] \tag{5}$$

If the two evolution rules F and F' are connected by this relation through T, then we say that F' is the T-transform of F and that the two models are connected by the transformation T.

The question now is whether the presence of a transformation connecting two models α and β ensures their equivalence.

4.2 Polynomial-time invertible transformations ensure equivalence

The flipping of the bits, described in Example 1, is actually a transformation, although quite simple. We expect that the models α and β, connected by such a transformation, are equivalent. Naïvely, we can expect that simple enough transformations ensure equivalence. A rigorous definition capturing the vague idea of simplicity is the following.

Definition 1 (Polynomial-time invertible transformation)**.** A transformation T is "polynomial-time invertible" if:

- T is invertible (i.e. one-to-one and onto), so that T^{-1} is defined;
- T can be calculated in polynomial time in the size of s;
- T^{-1} can be calculated in polynomial time in the size of s.

It can be easily seen that the flipping of the bits, described in Example 1, is actually a "polynomial-time invertible" transformation.

It is now possible to show that the presence of a polynomial-time invertible transformation T connecting two models α and β actually implies that they are equivalent, as stated in the following proposition.

Proposition 1. *We have two models α and β. If they are connected by a polynomial-time invertible transformation T, then they are equivalent.*

Proof. We must follow the description of the dispute discussed in Sect. 3.7, between Φ and Ψ, about the equivalence of two models α and β. Φ thinks that α and β are equivalent, while Ψ thinks that they are non-equivalent.

If Ψ proposes encoding and decoding functions for α, E_α and D_α, Φ can propose encoding and decoding functions for β, defined as follows:

$$E_\beta(p) = T\left[E_\alpha(p)\right] \tag{6}$$
$$D_\beta(s') = D_\alpha\left[T^{-1}(s')\right] \tag{7}$$

Since E_α, D_α, T, and T^{-1} operate in polynomial time, also E_β and D_β operate in polynomial time and are thus valid encoding and decoding functions. It is easy to see that, for whatever program p, the output of the execution in α with E_α and D_α is equal to the output of the execution in β with E_β and D_β.

On the other hand, we also see that Ψ cannot fulfill the requirement of point 2 of Sect. 3.7, also discussed in Sect. 3.6. Indeed, the output in α and β should correspond to the solution of two different instances e_1 and e_2 of an EXP-complete problem: it is impossible to calculate one from the other by means of a polynomial-time calculation. □

4.3 A (complex enough) transformation connects couples of models

If "simple enough" transformation ensure equivalence, are there "complex enough" transformations connecting non-equivalent models? The answer is likely positive. It is indeed possible to show that a transformation always exists between couples of models chosen in a particular set, which is however quite general. This transformation is defined below, but it requires a preliminaty definition.

Definition 2 (Model with counter)**.** A model has a "counter" if there is a function τ of the states with values in relative numbers such that:

$$\tau\left[F(s)\right] = \tau(s) + 1 \tag{8}$$

For whatever model, it is possible to obtain a "model with counter" by associating it to a register which is incremented at each time step.

Definition 3 (Brute-force transformation)**.** Consider two models α and β; α is a model with counter, with function τ. The "brute-force transformation" T between them is defined as:

$$T(s) = {F'}^{\tau(s)}\left[F^{-\tau(s)}\right] \tag{9}$$

It is easy to see that T transforms α into β.

Since the "models with counter" are a quite general type of models, it is likely that a couple of non-equivalent α and β can be found, where α is a model with counter. Such models are however connected by the "brute-force transformation". We can thus see that there can be transformations which do not ensure equivalence. .

5 Conclusions

I introduced the idea that the empirical equivalence can be studied from an "internal point of view" by numerically simulating a system, including a robot which performs experiments. I showed that a discussion about the equivalence of two models, from this "internal point of view", requires some subtleties and I suggested how to correctly perform it. The distinction between equivalent and non-equivalent models is based on the theory of computational complexity.

From the suggested procedure for discussing the equivalence, I obtained a sufficient condition for equivalence, i.e. the existance of a transformation between the two models that can be calculated in polynomial time. I gave examples of a transformation that meet this condition, and of a transformation that does not, and hence can transform a model into another model that is not equivalent.

Although the discussion only covers systems that are discrete (in space, time and states of cell), various properties of complexity classes have been extended to continuous systems. Whether this is possible in the present case will be the goal of future work.

References

[1] P. Duhem. *The Aim and Structure of Physical Theory.* Princeton University Press, Princeton NJ, 1954.

[2] C. Hoefer and A. Rosenberg. Empirical equivalence, underdetermination, and systems of the world. *Philosophy of Science*, 61:592–607, 1994.

[3] L. Laudan and J. Leplin. Empirical equivalence and underdetermination. *The Journal of Philosophy*, 88:449–472, 1991.

[4] P. Acuña and D. Dieks. Another look at empirical equivalence and underdetermination of theory choice. *European Journal for Philosophy of Science*, 4(2):153–180, 2013.

[5] P. Acuña and D. Dieks. *Artificial examples of empirical equivalence.* Springer, 2014.

[6] R. Boyd. Realism, underdetermination, and a causal theory of evidence. *Noûs*, 7:1–12, 1973.

[7] J. Worrall. Underdetermination, realism and empirical equivalence, 2011.

[8] P. Bak, C. Tang, and K. Wiesenfeld. Self-organized criticality: an explanation of $1/f$ noise. *Physical Review Letters*, 59(4), 1987.

[9] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach.* Cambridge University Press, New York, 2009.

[10] S. Aaronson. Why philosophers should care about computational complexity, 2011. manuscript Pre-print.

[11] Martin Gardner. The fantastic combinations of John Conway's new solitaire game of life. *Scientific American*, 223:120–123, 1970.

[12] P. Sarkar. A brief history of cellular automata. *ACM Computing Surveys*, 32(1):80–107, 2000.

[13] S. Wolfram. Universality and complexity in cellular automata. *Physica D: Nonlinear Phenomena*, 10(1–2):1–35, 1984.

[14] B. Durand and Zs. Róka. The game of life: universality revisited, 1999. Cellular automata, (Saissac, 1996) (M. Delorme and J. Mazoyer, eds.), Kluwer Acad. Publ., Dordrecht.

[15] M. Cook. Universality in elementary cellular automata. *Complex Systems*, 15(1):1–40, 2004.

[16] J. von Neumann and A. W. Burks. *Theory of self-reproducing automata*. University of Illinois Press, Urbana and London, 1966.

[17] C. H. Bennett. Logical reversibility of computation. *IBM Journal of Research and Development*, 17(6):525–532, 1973.

[18] A. M. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proc. London Math. Soc. Ser.*, 2(42):230–265, 1937.

[19] Nicolas Ollinger. Universalities in cellular automata; a (short) survey, 2008. Bruno Durand. JAC 2008, Uzès, France, Regular paper track. ¡hal-00274563¿ HAL Id: hal-00274563 https://hal.archives-ouvertes.fr/hal-00274563.

[20] Y. Rogozhin. Small universal turing machines. *Theoretical Computer Science*, 168(2):215–240, 1996.

[21] M. D. Davis and M. Davis. A note on universal turing machines. *Journal of Symbolic Logic*, 35(4):590–590, 1970.

[22] A. Gajardo and E. Goles. Circuit universality of two dimensional cellular automata: a review. In C. S. Calude, editor, *Randomness and Complexity: From Leibniz to Chaitin*, chapter 7. World Scientific Publishing, Singapore, 2007.

[23] N. Bostrom. Are we living in a computer simulation? *The Philosophical Quarterly*, 53(211):243–255, 2003.

[24] A. Anderson. Quantum canonical transformations. physical equivalence of quantum theories. *Phys. Lett. B*, 305(1–2):67–70, 1993.