Kairos. Journal of Philosophy & Science 24, 2020
Centre for Philosophy of Science of the University of Lisbon

Lying, computers and self-awareness

Paulo Castro
jpcastro@fc.ul.pt
Centre for the Philosophy of Science of the University of Lisbon

DOI 10.2478/kjps-2020-0009

Abstract: From the initial analysis of John Morris in 1976 about if computers can lie, I have presented my own treatment of the problem using what can be called a computational lying procedure. One that uses two Turing Machines. From there, I have argued that such a procedure cannot be implemented in a Turing Machine alone. A fundamental difficulty arises, concerning the computational representation of the self-knowledge a machine should have about the fact that it is lying. Contrary to Morris' claim, I have thus suggested that computers – as far as they are Turing Machines – cannot lie. Consequently, I have claimed that moral agency attribution to a robot or any other automated AI system, cannot be made, strictly grounded on imitating behaviors. Self-awareness as an ontological grounding for moral attribution must be evoked. This can pose a recognition problem from our part, should the sentient system be the only agent capable of acknowledging its own sentience.

Keywords: Lying, Philosophy of Lying, Turing Machine, computability, Artificial Intelligence (AI), moral agency, self-awareness, AI singularity, AI ethics.

1. Computers and the act of lying

Since its inception in Alan Turing's (1936) seminal paper «On Computable Numbers, with an Application to the Entscheidungsproblem», one could say that Computer Science has been for the most part, strictly concerned with the notion of truth making, thus constructing machines able to follow inferential processes. A computer can, in fact, be recognized as a truth producing machine, as initially meant by David Hilbert when he considered for the possibility of a hypothetical machine capable of formulating all mathematical theorems in an automatic fashion (Chaitin, 1997). It was only when the world revealed too complex to be computed, at least in a feasible time, that the symbolic approach gave way to the conexionist strategy. Such direction had already been proposed in the early forties, during the Macy Foundation Conferences, with the McCulloch-Pitt formal neuron. Following the same line of research, the Perceptron would be built later, at the beginning of the sixties, from the seminal work of Rosenblatt. With increasing technical maturity, the conexionist approach would further develop into the neural network technologies (Dreyfus, 1988) that constitutes AI actual trends.

Although truly promising, it must be emphasized that a neural network using rational numbers as its weight values is still computationally equivalent to a Turing Machine (Siegelmann, Sontag, 1995). Consequently, although the computational power of a neural network surmounts that of a Turing Machine, once real numbers are used, I think the arguments here will hold, since they have more to do with the algorithmic nature of artificial intelligence and not so much with their computational power.

Another very important remark concerns the adequacy of what will be suggested here, considering the recent work of Manuel, Elenore and Avrim Blum (2018). The authors have used Bernard Baars' Theather of Consciousness model (Baars, 1997), describing human conscious awareness, to design what they called the Conscious Turing Machine definition. The analogy with a stage act is quite straightforward: consciousness is defined as the content of what is being played in our short-term memory, understood as a stage. This staged content is then available to all unconscious (that is, out of the stage focus) processors in our brain. These processors are understood as a helping audience that sends further content to the stage to solve problems. The information sent by the processors to the short-term memory depend upon external sensorial input and whatever content was on stage, accordingly with some structural property or, somehow equivalently, encoded algorithm common to all unconscious processors. It is perhaps noticeable that the asserted definition of consciousness (what is in the stage) must include, in Manuel Blum own words (Blum, 2018), the existence of an inner speech – the faculty of talking to oneself. This is, in fact, what the acknowledgement of lying, underwent by a liar, entails: the unrolling of an inner communication, where the liar is permanently reminding himself that he is not telling the truth. It is this process, hypothetically performable by a Turing Machine, that the present paper tries to analyze. Since each attending processor in the Blums' model must comply to some algorithmic read/write strategy, as it reacts to its own inputs, I think the arguments I will be exposing here should also apply to the Blums's theoretical model of algorithmic consciousness.

So, let me start by formulating the problem.

In 1976, the philosopher John Morris offered his analysis for «If computers could ever lie» (Morris,1976). Although the question may seem of a strictly academic interest, I think it has a very serious contour we cannot afford to ignore. In fact, the problem may be said to be as old as Artificial Intelligence itself, in fact, being introduced even before Morris, in 1968, in Stanley Kubrick's movie "2001, A Space Odyssey", where HAL 9000 computer lies to David about the malfunctioning of an antenna. The movie has gathered a respected circle of enthusiasts in the academy, as with "HAL's Legacy" (Clarke, edited by Stork *ed.*, 1998), a book about the subject, where topmost computer scientists theorize about HAL's cognitive tasks, including its ability to feel emotions and, of course, to lie.

Consider the possibility that a computer has gained sentience, not at all paired to the human familiar one. First, we will not notice it. In fact, it has come to mind that the so-called singularity does not have to be a disruptive event in an instant in time. It can be otherwise a very long process, that has already begun, while no one was really noticing, simply because it is profoundly different from our own. And second, if so, when a machine would output the statement “I’m a machine” (as opposite to a sentient self-conscious organism), if it would be true, it would be just a statement, thus devoided of any intentionality, and if it would be false, it would be an assertion and therefore the machine would be lying. This implies in a certain sense that we, as technology producing agents, may not be in control of whatever may happen in the future when it comes to AI. If not because we do not understand what we have created, then because what we have created has harnessed the ancient art of lying and deceiving.

Contrary to Morris’ conclusion, my own analysis of the problem seems to suggest that a Turing Machine cannot, in fact, lie. I will claim that such capability also implies that the machine must be able to self-recognize that it is indeed lying. This self-recognition, according to the algorithmic representation of the act of lying that I will be presenting here, will produce a contradiction in the machine’s automated behavior. Of course, this only means that autonomous intelligence is perhaps more complex than a very long and intricated algorithm, and again, that what promotes or prevents AI may well be behind present technological comprehension. Hence becoming a problematic ethical endeavor at the very least, should sentience appear unexpectedly.

Traditionally defended by philosophers in the sixties and early seventies, the deceptionist definition of lying poses that to lie is to make a statement one believes to be false, while intending to produce in the listener the belief that it is true

(Meibauer, Mahon, 2018). However, such a plain definition may not be applicable to a computer since the machine may (or may not) lack a property one could easily equate to intentionality as a human agent normally would use it.

In his paper Morris removed the term ‘statement’ from the definition above and instead used the word ‘assertion’. He then introduced three modes of discourse implicit to the act of lying, in his own words:

> «1) The assertoric mode. The liar must assert something, for it to count as lying. Simply to mouth the words, as a person might do when reading from a book, is not to assert them, and therefore it cannot be lying.
> 2) The doxastic mode. The liar must believe something which contradicts his words. If he thinks that what he says is true, then he is not lying, even if it should turn out that he has spoken falsely; he may have been speaking ignorantly, hastily, or carelessly.
> 3) The volitional mode. The liar must actually want his listener to believe his words; this is somewhat similar to the requirement in law that, to be convicted of fraud, a person must be shown to have intended to defraud his victim» (Morris,1976: 390-391).

Of course, the volitional mode may be thought redundant once the assertoric mode has been stated. The reason is that when one asserts something, there is already an intention to cause belief in listeners, as assertionist philosophers would defend (1977, Chisholm, Feehan). I think Morris was thinking that to assert while lying implies the existence of a complex network of interlinked meanings, besides only wanting to cause belief in others, which he then considers a goal on its own, deserving to be stated as the assertoric mode.

Showing that a computer can lie is then to show, as Morris put it, that it can assert, believe, and want. I think this is a very tall order, considering what a Turing

Machine can do. Furthermore, it begs for very strong anthropomorphic interpretations about what it is for a machine to assert, believe or want. To overcome these problems, Morris placed the onus of identifying an assertion made by a computer on the external reader, rather than on some internal state of the machine. A printed statement would thus become such an assertion depending on the hermeneutical powers of the external reader, in some meaningful context and this type of criteria would then make it possible to say that a computer has asserted something and that it can assert. In effect, it can be said that it is exactly what happens when a human engages on a dialogue with a chatterbot or during a Turing test.

However, engaging on a critical analysis about the alleged human powers to judge about the machine capacities, I have shown elsewhere (Castro, 2017) that a Turing test relies so strongly on the authority role of the human judge, that once such authority is placed in question, the test becomes undecidable. To conclude then that a computer makes an assertion, depending only on the appreciation powers of an external cognitive agent can become an ambiguous act, to say the least. A stronger justification is needed.

As to the possibility of a computer wanting something, Morris plaid that a programmer can always embed a bias in the machine's behavior by means of its program thus, this becoming or embedding a sort of recorded volition. Again, I think that saying that a computer desires something, based on such simple criteria, seems philosophically naïve and begs for a stronger justification.

Finally, Morris holds that although a computer always believes in what it asserts (as it is strictly obeying an algorithm), it could find its way, as he argues, to dissemble «...the relationship between what it 'intends' to do and what it announces as its intentions, simulating the role of the volitional mode in a way that parallels the simulation of the doxastic mode» (Morris,1976, 398). It so happens that from what I will describe in the next sections, it will become clear that this dissembling process cannot occur in a Turing Machine.

2. When is a Turing Machine lying?

John Morris pioneering analysis seems to have missed the inner meaning of what a Turing Machine is. A very simply automaton that proceeds, step by step, only accomplishing whatever process is feasible in a mechanical way, independently therefore of the judgement powers of an external cognitive human agent.

To capture the meaning of the computational act of lying, we must therefore reappraise the problem of if computers can ever lie. We must ask, instead; in which circumstances would we say that a computer has lied or is lying to someone or to some other computer. In other words, we must find a formal acceptable definition for what can be acknowledged as the equivalent for the human act of lying, in a machine or, better and firstly, between machines.

We start by giving a comprehensive definition of a Turing Machine, that is; one that is equivalent to the usual mathematical one (Boolos, 1989), although adapted to our own purposes.

Definition 1, of a Turing Machine.

1.1. A Turing Machine is a conceptual mathematical scheme, representing the automated action of a machine that follows a set of instructions.

1.2. It is composed of a moving part P, representing the machine's processor, and a very long tape T, divided in equal cells, representing the machine's memory.

1.3. For each instruction to be carried on by the Turing Machine, the processor P can read, erase or to write a symbol at a time, to move in the right or left direction a cell at a time or to stand still.

1.4. The machine works using a finite set of symbols, called its alphabet Σ and its processor P can acquire a certain state from a finite set of states, called its state space S.

1.5. There is a symbol called the blank symbol, that stands for an empty cell when the machine is reading, and for an erasing action, when the machine is writing.

1.6. There is a finite set A of correspondence rules between the present state of the machine's processor P together with the symbol presently written in the tape and ready to be read and what the machine will do next (among the actions described in 1.3.), after reading the symbol in the tape. The set A is called the Turing Machine algorithm.

1.7. The Machine always acts accordingly to the following computing schema, respecting only the correspondence rules in the machine's algorithm.

$$R_S(a_1, S_1) = S_2 \qquad (1)$$

Meaning that if the Machine is in state and reads the symbol , it will acquire the state , accordingly to the correspondence rule in (1).

$$R_a(a_1, S_2) = a_2 \qquad (2)$$

Meaning that if the machine is in state (which it has already acquired, accordingly to (1)) and reads the symbol , it will write the symbol on the tape, accordingly to the correspondence rule in (2). We are simplifying such a procedure, by supposing that the act of writing automatically erases whatever symbol was written on the same cell of the tape, previously.

$$R_M(a_2, S_2) = M \qquad (3)$$

Meaning that if the machine is in state (which it has already acquired, accordingly to (1)) and reads the symbol (which it is already written on the tape, accordingly to (2)), it will execute a movement M, as defined in 1.3., accordingly to the correspondence rule in (3). We are simplifying such a procedure, by supposing that to stand still is also a possible movement M.

The machine repeats the computing schema, which we are denoting by:

$$R_S \rightarrow R_a \rightarrow R_M \qquad (4)$$

until it fails to find a correspondence rule in the algorithm A for its present state and for the symbol presently written on the tape. If such situation occurs, the machine stops, or halts and we say that the computation defined by its algorithm has been performed. We are including the cases where the Machine abruptly stops from our point of view, meaning that we have not written the correct algorithm for the intended purposes.

1.8. If the machine, to our best knowledge, is unable to stop, we say that the task handled using its algorithm is a non-computable one.

Since we now have a comprehensive operational definition of a Turing Machine, we ask in what situation should we say that a machine has computed a lie, in a similar way one would say a human has lied. To try and keep a certain objectivity, let us suppose that such a situation (as it would be by a human agent) can be recognized by another Turing Machine. The question is now in what situation can a Turing Machine decide that another Turing Machine is lying.

First, one may assume that the natural way to externally analyze if a Turing Machine is in fact lying, is to take its input data, output data and sequence of movements, and from there conclude something about the machine's truth production results. This is the mechanical analogy to the very human act of observing if someone is lying.

Before we can proceed, it is necessary to make some fundamental specification about the nature of the computational process that will be herein described. Initially, the case where a monitoring Turing Machine is analyzing the behavior of another Turing Machine, as to infer a lying behavior result, would apparently be analogous to the case where that same monitoring machine would be inductively identifying the algorithm used by the first machine. In fact, inferring that such algorithm would not be the appropriate one. However, I will defend shortly that the procedure that I am about to define, trying to capture when has a Turing Machine lied, is not an algorithmically inductive one, but a procedure that tests for the computational consistency of a given presented behavior. The test is done from the standpoint of the analyzing Turing Machine and, eventually, can be used as motivation to relate the concept of "belief" to the computational one of a "relative consistency". To see this, I will now give a very brief presentation of the inductive inference computational framework, returning to this point immediately after.

The first mathematical definition of an inductive Turing Machine was introduced by Mark Burgin in 1983 (Burgin, 1983, 2017). It can be regarded as the algorithmic mathematical implementation of the seminal logical models for limiting recursive and limiting partial recursive function (Gold, 1965) and for trial-and-error predicates (Putnam, 1965). Inductive inference has been theorized in the context of automatic learning of languages (Gold, 1964) and automatic discovery. The later, concerning the identification of a program that computes a given function graph or that performs the identification of a "law of nature" type of formula (or sets of such formulas) given, either sequences of empirical results or sets of possible theorems to be, and from which an initial axiomatic structure producing the former is inferred (Blum and Blum, 1975; Costa, 2017).

The computational spirit, if you will, of an automatized inductive procedure can be appreciated in the following definitions:

> «A class of problems is called decidable if there is an algorithm which will give the answer to any problem of the class after a finite length of time (...) [there are] classes of problems that can be solved by infinitely long decision procedures in the following sense: An algorithm is given which, for any problem of the class, generates an infinitely long sequence of guesses. The problem will be said to be solved in the limit if, after some finite point in the sequence, all the guesses are correct and the same (in case there is more than one correct answer) » (Gold,1965: 28).

And,

> «Informally, an inductive machine is an algorithm which is given larger and larger samples of the graph of a partial or total recursive function which the machine attempts to identify. The machine or algorithm produces at various stages a program which computes a recursive function. The machine is said to identify the target function if at some point it produces a program which computes that very function, and thereafter, no matter how much more of the graph of the target function it sees, continues to produce the same program. A weaker notion, that of behaviorally correct identification, does not require that to identify the target function the machine must converge to a single program. Instead, it requires only that the machine converges to a (possibly infinite) set of programs, all of which compute the target function» (Glymour, 1985: 23).

Although inductive Turing Machines are computationally more powerful than plain Turing Machines (Burgin, 2017, Syropoulos, 2010), it has been recognized from the study of mathematical inductive inference models that they have upper bound computational limits. Using Gold's terminology, «In general, subclasses of the class R of recursive functions [those calculated by an implementable algorithm in a machine] cannot be identified in the limit by recursive functions (Costa, 2017). Furthermore, if one now includes besides functions, collections of theorems to be interpreted as recursively enumerable sets (that is, of each its elements can be identified), then there is no computationally universal student that is able to identify (inductively) everything that is computationally identifiable. That is, each possible computational student can identify some sets and functions but not all sets and functions (Costa, 2017).

Now, getting back to what I intend to propose as a viable procedure effected by a watchful Turing Machine to identify when has a lie been produced by another, the overall strategy will be as follows. We will start by accepting that i) both machines can always act upon any common given set of symbols, using the same set of available states, the same alphabet of symbols and the same repertoire of movements, as described in Definition 1. This is a natural demand assuring that the machines are to communicate. Furthermore, it will also be accepted that ii) both machines will use the same type of correspondence rules in their algorithms, as defined in 1.7, although not necessary the same rules for all situations and, consequently, not always the same algorithms. Metaphorically, conditions i) and ii) mean that the machines represent cognitive agents sharing a common "computational Culture", if you will, expressing different views about the same topics.

To have a debunking Turing Machine identify the production of a lie by another Turing Machine, will imply firstly, that the former must act, accordingly to its own correspondence rules, upon the same set of symbols inputted and outputted by the later. Secondly, it will also imply that the debunking Machine must compare the algorithm it has used to compute the symbols, inputted, and outputted by the lying Turing Machine, with the algorithm used by the later. If both diverge, the debunking Turing Machine will decide for the "untruthfulness" of the procedure effected by the lying machine.

From what has just been said, the main issue in the suggested procedure is then not to identify (in a algorithmically inductive way or otherwise) what was the program (set of correspondence rules) that was used by the lying Turing Machine to produce its computation, but rather to assert, at some point of that computation, that the algorithm used by the lying Turing Machine is different from the one used by the debunking machine (upon the same set of symbols) and hence that the former must be lying.

Of course, in this framework, such a declaration implies that truthfulness (in a quite abstract sense) must be arbitrarily (that is, in a manner external to the devices) attributed to the debunking Turing Machine and making its "algorithmic point of view" the authoritative one. This invokes the question of asserting the weight social accordance may have on selecting between what is commonly believed to be true and what is not. As it will become clear, the debunking Turing Machine will perform a kind of relative consistency check of the algorithm used by the lying Turing Machine. It will do so against its own algorithmic knowledge and grounded on its authority stand.

Finally, a word about the algorithmic capabilities of an inductive Turing Machine, performing as a debunking machine. To assert that the lying Turing Machine is not telling the truth, the inductive machine output results would naturally have to diverge (in a finite number of steps) from the ones produced by the algorithm followed by the lying machine. Since an inductive Turing Machine starts its procedure by approaching the correct output, in general, from an initial different one, it will always (wrongly) identify a lie in a few steps of its computation.

In other words, in the scheme suggested here, the inductive computational powers of a debunking inductive Turing Machine will not be adequate to perform the lying identification task, as it will perform poorly than a standard Turing Machine. Hence, to perform as a debunking Turing Machine, an inductive Turing Machine must act as a simple Turing Machine, which means that the objections that will be posit concerning simple Turing Machines abilities to lie will also apply to inductive Turing Machines.

I now proceed to describe concretely what an analyzing Turing Machine must do in search for evidence that another machine has lied.

For a debunking Turing Machine dTM to perform an analysis on the truth behavior of a possible lying Turing Machine lTM, let us suppose that a device can be coupled to lTM verifying the following conditions:

i) Whenever lTM reads a symbol from its own tape T, that symbol is written in a second tape T'.
ii) Whenever lTM, writes a symbol in its own tape T, that symbol is written in a second tape T'.
iii) Whenever lTM makes a movement M along its own tape, that movement is correspondingly represented by a written symbol in a second tape T' (e.g., R, for right, L for left and N for no movement).
iv) All symbols are written in T' in the same sequence as the one taken by lTM during its operation accordingly to the computation schema described in 1.7.
v) T' is the input tape to be analyzed by the debunking Turing Machine dTM.

In the following we will accept, as a matter of principle, that the dTM can be programmed to proceed as it will be suggested. As we will see, the intended procedures are quite reasonable for a Turing Machine to perform. Furthermore, perhaps more stringently, we will demand the dTM to be able to output, in some suitable form of coding, the internal states representation within the correspondence rules in its own algorithm.

Let us say the dTM has inputted the same initial symbol , as the lying Turing Machine lTM. Now, according to its own algorithm, and following the computing schema described in 1.7., lTM outputs the symbol . The watchful dTM inputs the newly written symbol from T', and promptly looks within its own algorithm for a correspondence rule, verifying:

$$R_a(a_1, S'_1) = a_2 \tag{5}$$

If dTM encounters such a rule, it will of course infer that lTM's initial state was.

Still following its own algorithm, and having outputted the symbol , lTM now exhibits a movement M. Again, the watchful dTM inputs from the tape T' the corresponding symbol , and looks within its own algorithm for a correspondence rule, verifying:

$$R_M(a_2, S''_1) = M \tag{6}$$

From this last rule, dTM infers that the lying Turing Machine's initial state was .

At this point, it becomes clear that the only way to "catch a lie", so to speak, in this situation, is to compare states and . Since, according to the computing schema described in 1.7, the initial state of lTM can only be one and the same, and if and are two possible values for the same initial state in (5) and (6), then they must necessarily be equal. Hence, if they are different, we will say that the analyzed

Turing Machine has computed a lie or, for short, that it has lied. We can thus, write in a more systematic way:

Definition 2, of when a Turing Machine has lied.

Given an analyzable Turing Machine and a debunking Turing Machine, given that i) the two machines can always act upon any common given set of symbols using the same set of available states, the same alphabet of symbols and the same repertoire of movements, as described in Definition 1 and that ii) both machines will also use the same type of correspondence rules in their algorithms, as defined in 1.7, although not necessary the same rules for all situations and, consequently, not always the same algorithms, the first is said to have computed a lie or to have lied to the second, if the debunking Turing Machine, from the behavior of the first, has outputted an expression asserting the fact that it was not able to find any correspondence rules, in its own algorithm, for which:

$$S'_1 = S''_1 \qquad (7)$$

Where is the initial state of the analyzable Turing Machine, as defined in 1.7

This definition calls for the following remark about consistency checking.

One might be led to believe that as the debunking Turing Machine has identified an inconsistency in the lying Turing Machine procedure which would therefore imply that the later could not have computed anything, in the first place. However, this is not true since such consistency appraisal is not absolute in a universal computational sense (the debunking Turing Machine is not a universal algorithm that checks for consistency in any conceivable situation). Its verification proceeds only in relation to the debunking machine's own algorithmic knowledge.

What happened was that the lying Turing Machine used a different algorithm to perform the computation as observed by the debunking Turing Machine. The lying Machine used other correspondence rules, different from the ones found by dTM to be the adequate ones for the inputted and outputted symbols, let us say:

$$R'_a(a_1, S'''_1) = a_2 \tag{8}$$

And

$$R'_M(a_2, S'''_1) = M \tag{9}$$

Consequently, the debunking machine identified what can be called an erroneous procedure only once compared to the algorithmic information contained in its own algorithmic data bank. Once more, the discussion evokes the very important philosophical problem of knowing from whom, where or what emanates the authority to define what is or is not true. I will not dwell with that difficulty here. I will simply accept that in the given situation, the debunking Turing Machine proceeded in a possible acceptable and truthful way.

Definition 2. tries to capture the inner spirit of a lie in so much that the lines of thought of the liar and of the receiver, in relation to the same narrative (in the case, the symbols read and written, and the movement of the machine), are different. To lie and to ear a lie is therefore, as far as this definition goes, an exercise of relativeness. As already noted, one which, most critically, depends on the fact that only one Turing Machine has performed truthfully. In this manner, it can be said that the computational act of lying corresponds to the existence of different algorithms, running in two, otherwise, equal Turing Machines, where only one of them authoritatively encloses the truth.

3. Self-deception and computability

Since we have found an effective procedure, making it possible for a Turing Machine to identify a lie from the output behavior of another Turing Machine, it seems that Turing Machines can, in fact, lie. However, there is more to it.

In the first place, the above procedure can be easily equated with the situation where two cognitive agents disagree one with the other. The same operatorial definition could be given for a disagreement identification, performed by Turing Machine, while analyzing another.

But even if, at a basic computational level, to lie could be characterized as a particular kind of disagreement between two cognitive agents, there is a major difficulty that, I think, makes lying something very different from a purely mechanical act. The problem begins with the remark that when one lies, one seems to be aware that one is lying. The liar knows that he is lying, and this has profound logical implications for a Turing Machine.

To start arguing about this point, I take the act of self-deception, which is a situation where a person tries to lie to oneself, denying something that the same person does not wish to acknowledge. Although this situation is different from lying to someone else, I think it rather illustrates the fact that a sort of self-monitoring process is at play when a human cognitive agent lies. I am, thus, suggesting that the same mental process of self-monitoring is always active when someone lies, either to himself or to others. The only difference being that in the act of self-deception, the self-monitoring process may be easier to identify by an external observer. In the paper «On the Psychology of Self-Deception» David Shapiro notes, in fact, that:

> «Self-deception can easily seem paradoxical. How can the knowing deceiver also be the unknowing deceived? How can one intentionally, knowingly, not know? The process clearly requires a selective monitoring of oneself, and that selectiveness is usually taken to imply both knowing

> what must not be known and at the same time being able not to know it. » (Shapiro, 1996, 786)

About someone who is self-deceiving himself, he also says that:

> «The speaker's voice is often louder than his normal conversational voice. He does not seem to be looking at one in the ordinary way. The listener does not seem to be in his focus; he seems to be looking past him. One feels tempted to wave one's hand to catch the speaker's attention. His attention seems inward, in the way of someone listening to himself, like a person who is practicing a speech (…) He is addressing himself through the listener. » (Shapiro, 1996, 789-790)

That a self-monitoring process is, in fact, active, when a human cognitive agent lies, poses an unsurmountable difficulty for a Turing Machine, for, consider the following. If such a process would be active in the machine, that would mean that the machine would have to acknowledge both what is the truth and what is not. This simply accounts to apply the procedure leading to definition 2, of when a Turing Machine has lied, to the machine itself. Let us do just that and see what happens.

Now the debunking Turing Machine dTM coincides with the lying Turing Machine. So, again, let us say the machine inputs the initial symbol (from its own tape). According to its own algorithm, and following the computing schema described in 1.7., the automata outputs the symbol . Then, it inputs the newly written symbol and promptly looks within its own algorithm for a correspondence rule, verifying:

$$R_a(a_1, S'_1) = a_2 \qquad (10)$$

It obviously encounters such a rule, because it has computed thus far, using as its input and outputting . That is, the Turing Machine finds the rule:

$$R'_a(a_1,S'''_1) = a_2 \tag{11}$$

From this it infers that its initial state was and, of course:

$$R_a \equiv R'_a \tag{12}$$

Meaning that both the searched correspondence rule and the one found are the same rule. This implies:

$$S'_1 = S'''_1 \tag{13}$$

Proceeding with its own algorithm, and having outputted the symbol , the machine makes a movement M. Let us suppose that whenever it moves, the machine writes the proper representative symbol . The automaton now inputs it, and looks within its own algorithm for a correspondence rule, verifying:

$$R_M(a_2, S''_1) = M \tag{14}$$

Evidently, it finds it, since the machine has just moved accordingly, after having inputted the symbol . Hence, the Turing Machine finds the rule:

$$R'_M(a_2, S'''_1) = a_2 \tag{15}$$

From this last rule, the Turing Machine infers that its initial state was , and again:

$$R_M \equiv R'_M \tag{16}$$

Meaning that both the searched correspondence rule for movement and the one found are the same rule. Once more, this implies:

$$S''_1 = S'''_1 \tag{17}$$

Which, finally, leads to:

$$S'_1 = S''_1 \tag{18}$$

If we now recall Definition 2, we conclude that for a Turing Machine to be lying, condition (18) must be false, which is obviously not the case. Consequently, we have concluded, using Definition 2, that a Turing Machine cannot possibly output that it is, itself, lying. We will rephrase this by loosely saying that a Turing Machine cannot have any knowledge that it is lying, while it is lying. In the present framework, lying awareness and self-deception, therefore, seem to be non-computable actions for a Turing Machine.

4. Conclusions

So, can a computer lie? That is, can a Turing Machine lie? Based on what we have reasoned, the answer must be no. To lie, a cognitive agent must have the knowledge that he is lying. Not only, because, phenomenologically, that is what we assert, but also because an agent can only become morally accountable for his own lie - and thus, truly, a liar - if he knows that he is lying. It is the effectiveness of such knowledge that makes a lying agent become a moral agent.

Presently, there is deep concern about the moral obligations and responsibilities attributable to automaton behavior (Gunkel, 2017). When an industrial robot kills a man or when an automated car runs over a person, who is to blame, the machine's constructor, its users, or the machine? Does intelligent autonomy in a machine wrap entirely the question or should we demand self-awareness as the ontological basis from which we can say that a robot has become morally accountable? And how should we define self-awareness in a machine, if it happens to be a kind of non-computable property, at least, in the Turing Machine sense?

I think most of these questions about the machines ethical framing, brought about by AI, are presently associated with two common ground beliefs coming from tradition. First, that the further away a living organism is from a human being, the lesser it has moral rights, duties or, even, any ethical dignity, as comparable to us. Second, that every living organism is, in fact, also a machine. I think AI developers, hence, see robots to be just imitating machines devoided of any moral agency. That is, devices deprived of any organicity. In short, on one hand, they are only mechanically imitating living systems and, on the other, they are sufficiently far from us to be "justly" deprived of any ethical dignity.

The problem, of course, is that such beliefs may be plain wrong. Animal ethics surely may shed some doubt about the ethical dignity issue (Singer, 2001) and, for the second, it is not at all clear at what level of complexity does a machine become a living organism.

Considering that a computer cannot lie, I suggest that moral agency cannot be attributed to a robot or any other automated AI system, based only on its imitating skills of human behavior. We must instead take into account what I am also suggesting to be a non-computable property – in the Turing Machine sense – that is, the property of self-awareness. The problem, however, is that this property seems to be only recognizable from an inner phenomenological point of view. That is, only the self-aware being knows that he is self-aware. If this is true, we risk missing moral attribution to a sentient robot, one that has transcended its computational limitations.

The so-called AI singularity has been supposedly considered to be a disruption in historical time, corresponding to some enlightening isolated moment in computer science.

On the contrary, I suggest that such singularity may be the result of an evolutionary process, that has already begun with the aid of human intervention, storing unpredictable results in the future. As such, the act of lying may then be thought as a possible signature of a new phenomenological state in a robot's mind, so-to-speak. That is, the signature of one possible high-level behavior that a sentient organism may exhibit.

Acknowledgments

I would like to thank the anonymous referees for the comments and criticism which helped improve this presentation. This work has been funded by FCT (Portugal) through CFCUL (project UID/FIL/00678/2019).

References

Baars, B. J. (2001). In the Theater of Consciousness: The Workspace of the Mind (New Ed edition). Oxford University Press.

Blum, M. (2018, August 31). Towards a Conscious AI: A Computer Architecture Inspired by Neuroscience [Text]. EECS at UC Berkeley. https://eecs.berkeley.edu/turing-colloquium/schedule/blum

Blum, L., & Blum, M. (1975). Toward a Mathematical Theory of Inductive Inference. *Inf. Control.*, 28, 125-155.

Boolos, G., &. Jeffrey, R. (1989). *Computability and Logic*. 3rd ed. Cambridge University Press.

Burgin M (1983) Inductive Turing machines. Not Acad Sci USSR 270(6):1289–1293. translated from Russian, v. 27, No. 3

Burgin, M. (2017) Inductive Turing Machines. In R. A. Meyers (Ed.), Encyclopedia of Complexity and Systems Science (pp. 1–14). Springer.

Castro P. (2017). «Computing Machinery, Intelligence and Undecidability». *Journal of Theoretical & Computational Science* 4, 160.

Chaitin, G. (1997). *The Limits of Mathematics: A Course on Information Theory and the Limits of Formal Reasoning*. National University of Singapore,Singapore.

Chisholm, R., & Feehan, T. (1977). «The Intent to Deceive». *The Journal of Philosophy*, 74(3), 43-159.

Clarke, A. C. (1998). HAL's Legacy: 2001's Computer as Dream and Reality (D. G. Stork, Ed.; Reprint edition). The MIT Press.

Costa, J. F. (2017) Unity of Science as Seen Through the Universal Computer, *International Journal of Unconventional Computing*, issue 13.1, p. 59-81,

Dreyfus, H., Dreyfus, S. (1988). «Making a Mind Versus Modeling a Brain: Artificial Intelligence Back at a Branchpoint». In *The Artificial Debate: False Starts, Real Foundations*, 15-43. MIT Press.

Glymour C. (1985) Inductive Inference in the Limit. In: Essler W.K., Putnam H., Stegmüller W. (eds) Epistemology, Methodology, and Philosophy of Science. Springer, Dordrecht

Gold, E. Mark, Language identification in the limit. Santa Monica, CA: RAND Corporation, 1964.

Gold, E. (1965). Limiting recursion. *Journal of Symbolic Logic*, 30(1), 28-48.

Gunkel, D. J. (2017). *The Machine Question: Critical Perspectives on AI, Robots, and Ethics*. The MIT Press.

H.T. Siegelmann and E.D. Sontag. (1995). "Computational Power of Neural Networks," Journal of Computer System Sciences, 50(1): 132-150

Meibauer, J., & Mahon, J. (2018-11-15). «Contemporary Approaches to the Philosophy of Lying». In The Oxford Handbook of Lying: Oxford University Press. Jan. 2020

Morris, John. (1976) «Can computers ever lie?» *World Futures*, 14 (4): 389-401

Putnam, H. (1965). Trial and error predicates and the solution to a problem of Mostowski. *Journal of Symbolic Logic*, 30(1), 49-57.

Shapiro, D. (1996). «On the Psychology of Self-Deception». *Social Research*, 63(3), 785-800.

Singer, P. (2001). *Writings on an Ethical Life*. Harper Perennial.

Syropoulos, A. (2010). Hypercomputation: Computing Beyond the Church-Turing Barrier. Springer.

Turing, A. (1936). «On Computable Numbers, with an Application to the Entscheidungsproblem». *Proceedings of the London Mathematical Society. Second Series,* 1936.