NON-COMPUTABILITY OF HUMAN INTELLIGENCE

YASHA SAVELYEV

Abstract. We revisit the question (most famously) initiated by Turing: can human intelligence be completely modeled by a Turing machine? We show that the answer is *no*, assuming a certain weak soundness hypothesis. More specifically we show that at least some meaningful thought processes of the brain cannot be Turing computable. In particular some physical processes are not Turing computable, which is not entirely expected. There are some similarities of our argument with the well known Lucas-Penrose argument, but we work purely on the level of Turing machines, and do not use Gödel's incompleteness theorem or any direct analogue. Instead we construct directly and use a weak analogue of a Gödel statement for a certain system which involves our human, this allows us to side-step some (possible) meta-logical issues with their argument.

We study the following question:

Question 1. Can human intelligence be completely modelled by a Turing machine?

We will give a complete definition of a Turing machine after the introduction. An informal definition of a Turing machine [1] is as follows: it is an abstract machine which accepts certain inputs, and produces outputs. The outputs are determined from the inputs by a fixed finite algorithm, in a specific sense.

In particular anything that can be computed by computers as we know them can be computed by a Turing machine. For the purpose of the main result the reader may simply understand a Turing machine as a digital computer with unbounded memory running a certain program. Unbounded memory is just mathematical convenience, it can in specific arguments, also of the kind we make, be replaced by non-explicitly bounded memory.

Turing himself has started on a form of Question 1 in his Computing machines and Intelligence, [2], where he also outlined a possible obstruction to a yes answer coming from Gödel's incompleteness theorem. He pointed out that one way to avoid this obstruction is to reject the assumption that humans are fundamentally consistent. What the latter actually means in practice is subject to a lengthy discussion, we need some qualifier like "fundamental" as even mathematicians do not on the surface assert consistent statements at all times.

There are a number of ways of interpreting Question 1. Turing himself was mainly interested in whether a Turing machine can fool an experimenter into believing that it is a human subject, in various Imitation Games, see [2] for examples. As this author understands, Turing believed roughly the following, (here for exposition's sake we take the liberty of compressing, perhaps not completely accurately, Turing's ideas into a neat hypothesis):

Hypothesis 1 (Turing's hypothesis)**.** *For every given human experimenter, an imitation game, and given some bounded amount of time, one can construct a Turing machine that will fool him in this imitation game, for that amount of time.*

This particular hypothesis may well be true, nothing in the present note contradicts it. However our interpretation of the question is different. We are asking is whether there is a meaningful mathematical difference in operation of a human mind and a Turing machine. While the significance of this version of the question for computer science is perhaps arguable, for physics and biology it is profoundly important.

It should be pointed out that a common misconception is that there can be no such mathematical difference if one believes the universe to be governed by deterministic, or effectively for a given purpose deterministic, laws. Quantum mechanics for instance is effectively for our purpose deterministic, as

the results (measurements) are at worst given by a probability distribution that is determined. But deterministic does *not* imply computable in the mathematical sense, e.g. Turing computable, (this may refer also to the above mentioned probability distribution). The latter would be computably deterministic, and this is the source of the misconception. [1]

Gödel himself first argued that such a mathematical difference exists, [3, p. 310]. Later Lucas [4] and later again and more robustly Penrose [5] strongly argued that a meaningful difference exists and for a no answer to question 1. They further formalized and elaborated the obstruction coming from Gödel's incompleteness theorem. And they reject the possibility that humans could be inconsistent on a fundamental level. The main common objections to their argument concern the meta-logic of the argument, see for instance [6]. We shall say a few words about this below. The other objection is simply that the consistency assumption may be wrong, this is suggested by Turing himself for instance as mentioned above. For a discussion of the consistency question we refer the reader to the wonderful books of Penrose, [7], [5].

It should also be noted that for Penrose in particular, non-computability of intelligence is evidence for new physics, and he has specific and *very* intriguing proposals with Hameroff [8], on how this can take place in the human brain. Another physical argument for non-computability is presented in Song [9]. Here is a partial list of some partially related work on mathematical models of brain activity and or quantum collapse models: [10], [11], [12], [13].

We likewise argue here for such a mathematical difference. Like Lucas-Penrose we have a soundness assumption but our seems much weaker. We do not use Gödel incompleteness theorem, instead we construct some weak analogue of Gödel statement directly. The following is a slightly informal version of our main Theorem 2.

Theorem 1. *The answer to Question 1 is no, assuming a certain soundness condition on our humans. More specifically, at least some human being S as a component of a simple certain physical system, cannot be both Turing computable and sound in a specific sense.*

In what follows we refer to some language of abstract formal systems but none of this appears in the main argument, as our language will be purely that of Turing machines. It will be instructive to first give a preliminary version of Penrose argument, in our language, and see what goes wrong. But to emphasize this is *not* his argument.

Let P be a human subject, which we understand as a machine printing statements in arithmetic, given some input. That is for each Σ some string input, $P(\Sigma)$ is a statement in arithmetic, e.g. Fermat's last theorem. Say now P is in contact with experimenter/operator E. The input string Σ that E gives P is the following: "here is the specification of a Turing machine T, this machine computes you as a (perhaps partially defined) function $\Sigma \mapsto P(\Sigma)$". Before we proceed, we put the condition on our P that he believes himself to be fundamentally sound, that is he believes $P(\Sigma)$ is true for all Σ. This is not an assumption by P, this is what P asserts as *truth*, whether this is actually rational is another discussion as already mentioned. Now P replies:

> *I know I am sound, hence T must be sound if it did compute me, but I can then construct a Gödel statement $G(T)$ for T, which is then a true statement in arithmetic, and which indirectly asserts "T cannot print $G(T)$".*

This alone doesn't quite work however for E counters:

> *I see, but do* you *in fact in print $G(T)$? This is the only way you can presently reach a contradiction, as T only computes what you print, not what you can meta-prove.*

P has to say no! Because he does not in fact know $G(T)$ is true, since he has to know that T is sound, and this only happens (from his point of view) if he knew for a fact: T computes P. If P does print $G(T)$ then he is patently unsound, as he has no basis to assert $G(T)$, so his very belief that he is sound would be absurd.

[1] We should mention that there is also a notion of non-deterministic Turing machines but this "non-deterministic" is not directly related to our discussion. Moreover these machines can be simulated by Turing machines of the type we consider, so are not considered here.

0.0.1. *A possible fix.* (Outline) We can try to fix the above argument as outlined in the following, and this "fixed version" will be very close in essence to the argument Penrose gives in [7], which we take to be his main and final argument, (as far as I am aware).

Change Σ to: "here is a specification of a Turing machine T, print your statement that you assert to be true assuming Θ_T:

$$T \text{ computes } P." \tag{0.1}$$

Again P reasons that since his deductions based on Θ_T are sound, the same must be true of T. Then (perhaps) there is a conditional Gödel statement $G(T, \Theta_T)$, which is true if $\mathcal{F}(T) + \Theta_T$ is sound and indirectly asserts that T cannot print $G(T, \Theta_T)$. Here $\mathcal{F}(T)+\Theta_T$ denotes the formal system underlying T adjoined with the assertion Θ_T as an axiom. P then prints $G(T, \Theta_T)$.

Now if Θ_T is true, and P is indeed sound and hence T is sound, then T cannot print $G(T, \Theta_T)$. So we reach a contradiction, unless P is not in fact sound. The above outline is still not totally satisfactory. For one we need to show that conditional Gödel statements as above can be constructed (by P). At the least this requires that Θ_T be interpreted as a statement in a suitable formal system (containing arithmetic). For this we need P to be suitably defined, otherwise it is not clear how the interpretation of Θ_T as a statement could work, but we delve no further. See however [14], and [6] for discussions on related issues.

This note can be understood as an elaboration of the above "fixed" argument, but with additional changes and improvements. First we do not need our subject S to be totally sound, we only need soundness of a certain much more limited function, associated to S. Moreover we do not need S to explicitly construct Gödel statements, for we find in this context an elementary and explicit analogue, a kind of weak Gödel statement directly, and this suffices for S. Thus Gödel incompleteness theorem is not used at all. Note also that we formalize at least partly some necessary properties of S or rather of a certain system encompassing S. This is in contrast with P above being essentially undefined. Finally we use exclusively the language of Turing machines, as opposed to formal systems, the former is vastly more elementary and concrete. In particular the above mentioned weak Gödel statement is formulated purely in the language of Turing machines.

1. Some preliminaries

This section can be just skimmed on a first reading. Really what we are interested in is not Turing machines per se but computations that can be simulated by Turing machine computations. These can for example be computations that a mathematician performs with paper and pencil, and indeed is the original motivation for Turing's specific model. However to introduce Turing computations we need Turing machines, here is our version, which is a computationally equivalent minor variation of Turing's original machine.

Definition 1.1. *A* **Turing machine** M *consists of:*

- *Three infinite (1-dimensional) tapes T_i, T_o, T_c, divided into discreet cells, one next to each other. Each cell contains a symbol from some finite alphabet. A special symbol b for blank, (the only symbol which may appear infinitely many often).*
- *Three heads H_i, H_o, H_c (pointing devices), H_i can read each cell in T_i to which it points, H_o, H_c can read/write each cell in T_o, T_c to which it points. The heads can then move left or right on the tape.*
- *A set of internal states Q, among these is "start" state q_0. And a non-empty set F of final, "finish" states.*
- *Input string Σ, the collection of symbols on the tape T_i, so that to the left and right of Σ there are only symbols b. We assume that in state q_0, H_i points to the beginning of the input string, and that the T_c, T_o have only b symbols.*
- *A finite set of instructions I that given the state q the machine is in currently, and given the symbols the heads are pointing to, tells M to do the following, the taken actions 1-3 below will be (jointly) called an* **executed instruction set***, or just* **step***:*

(1) *Replace symbols with another symbol in the cells to which the heads H_c, H_o point (or leave them).*
(2) *Move each head H_i, H_c, H_o left, right, or leave it in place, (independently).*
(3) *Change state q to another state or keep it.*

- *Output string Σ_{out}, the collection of symbols on the tape T_o, so that to the left and right of Σ there are only symbols b, when the machine state is final. When the internal state is one of the final states we ask that the instructions are to do nothing, so that these are frozen states.*

We also have the following minor variations on standard definitions, and notation.

Definition 1.2. *A* **complete configuration** *of a Turing machine M or* **total state** *is the collection of all current symbols on the tapes, position of the heads, and current internal state. A* **Turing computation**, *or* **computation sequence** *for M is a possibly not eventually constant sequence*

$$\{s_i\}_{i=0}^{i=\infty} := *M(\Sigma)$$

of complete configurations of M, determined by the input Σ and M, with s_0 the initial configuration whose internal state is q_0. If elements of $\{s_i\}_{i=0}^{i=\infty}$ are eventually in some final machine state, so that the sequence is eventually constant, then we say that the computation **halts**. *For a given Turing computation $*M(\Sigma)$, we shall write*

$$*M(\Sigma) \to x,$$

if $*M(\Sigma)$ halts and x is the output string.

We write $M(\Sigma)$ for the output string of M, given the input string Σ, if the associated Turing computation $*M(\Sigma)$ halts.

Definition 1.3. *Let Strings denote the set of all finite strings of symbols in some fixed finite alphabet, for example $\{0, 1\}$. Given a partially defined function f : Strings $\to$ Strings, that is a function defined on some subset of Strings - we say that a Turing machine M* **computes** *f if $*M(\Sigma) \to f(\Sigma)$, whenever $f(\Sigma)$ is defined.*

For writing purposes, let us call a partially defined function f : *Strings* $\to$ *Strings* as above an ***operator***, and write $\mathcal{O}$ for the set of operators. So a Turing machine T itself determines an operator, which is defined on all $\Sigma \in$ *Strings* s.t. $*T(\Sigma)$ halts, by $\Sigma \mapsto T(\Sigma)$.

The following definition is also purely for writing purposes.

Definition 1.4. *Given Turing computations (for possibly distinct Turing machines) $*T_1(\Sigma_1)$, $*T_2(\Sigma_2)$ we say that they are* **equivalent** *if either they both halt with the same output string, or both do not halt. We say a pair of Turing machines T_1, T_2 are equivalent if they compute the same operator.*

Let us expand the above discussion a bit. We will allow our Turing machine T to reject some elements of *Strings* as valid input. We may formalize this by asking that there is a special final machine state q_{reject}, so that $T(\Sigma)$ halts with q_{reject} for $\Sigma \notin I \subset$ *Strings*, where I is some set of all valid, that is T-***permissible*** input strings. Note we do not ask that for $\Sigma \in I$, $*T(\Sigma)$ halts. If $*T(\Sigma)$ does halt then we shall say that Σ is ***acceptable***. It will be convenient to forget q_{reject} and instead write $T : I \to O$, where $I \subset$ *Strings* is understood as the subset of all T-permissible strings, and O is the set output strings, keeping all other data implicit. The specific interpretation should be clear in context.

We also note that all of our input, output sets are understood to be subsets of *Strings* under some encoding. For example if the input set is $Strings^2$, we may encode it as a subset of *Strings* via encoding of the type: "this string Σ encodes an element of $Strings^2$ its components are Σ_1 and Σ_2." In particular the sets of integers $\mathbb{N}, \mathbb{Z}$ will under some encoding correspond to subsets of *Strings*. However it will be often convenient to refer to input, output sets abstractly without reference to encoding subsets of *Strings*. (Indeed this is how computer languages work.)

Remark 1. The above elaborations mostly just have to do with minor set theoretic issues. For example we will want to work with some "sets" $\mathcal{T}$ of Turing machines, with abstract sets of inputs and outputs.

These "sets" $\mathcal{T}$ will truly be sets if implicitly all these abstract sets of inputs and outputs are encoded as subsets of *Strings*.

Definition 1.5. *We say that a Turing machine T computes an operator $f : Strings \to Strings$ on $A \subset Strings$ if A is contained in the subset I of T-permissible strings, and $*M(\Sigma) \to f(\Sigma)$, whenever $f(\Sigma)$ is defined, for $\Sigma \in A$.*

Given Turing machines $M_1 : I \to O$, $M_2 : J \to P$, where $O \subset J$, we may naturally ***compose*** them to get a Turing machine $M_2 \circ M_1$, let us not elaborate as this should be clear, we will use this later on.

1.1. **Join of Turing machines.** There is a simple variant of a Turing machine where a single tape is replaced by a multi-tape. Indeed our Turing machine of Definition 1.1 is a multi-tape version of a more basic notion of a Turing machine with a single tape, but we need to iterate this further.

We replace a single tape by tapes $T^1, \dots, T^n$ in parallel, which we denote by $(T^1 \dots T^n)$ and call this n-tape. The head H on the n-tape has components H^i pointing on the corresponding tape T^i. When moving a head we move all of its components separately. A string of symbols on $(T^1 \dots T^n)$ is an n-string, formally just an element of $\Sigma \in Strings^n$, with i'th component of Σ specifying a string of symbols on T^i.

Given Turing machines M_1, M_2 we can construct what we call a ***join*** $M_1 \star M_2$, which is roughly a Turing machine where we alternate the operations of M_1, M_2. In what follows symbols with superscript with $1, 2$ denote the corresponding objects of M_1, respectively M_2, cf. Definition 1.1.

$M_1 \star M_2$ has three (2)-tapes:

$$(T_i^1 T_i^2), (T_c^1 T_c^2), (T_o^1 T_o^2),$$

three heads H_i, H_c, H_o which have component heads H_i^j, H_c^j, H_o^j, $j = 1, 2$. Machine states:

$$Q_{M_1 \star M_2} = Q^1 \times Q^2 \times \mathbb{Z}_2,$$

with initial state $(q_0^1, q_0^2, 0)$ and final states:

$$F_{M_1 \star M_2} = F^1 \times Q^2 \times \{1\} \sqcup Q^1 \times F^2 \times \{0\}.$$

Then given machine state $q = (q^1, q^2, 0)$ and the symbols $(\sigma_i^1 \sigma_i^2), (\sigma_c^1 \sigma_c^2), (\sigma_o^1 \sigma_o^2)$ the heads H_i, H_c, H_o are currently pointing, to we first check instructions in I^1 for q^1, $\sigma_i^1, \sigma_c^1, \sigma_o^1$, and given those instructions as step 1 execute:

(1) Replace symbols σ_c^1, σ_o^1 to which the head components H_c^1, H_o^1 point (or leave them in place, the second components are unchanged).
(2) Move each head component H_i^1, H_c^1, H_o^1 left, right, or leave it in place, (independently). (The second component of the head is unchanged.)
(3) Change the first component of q to another or keep it. (The second component is unchanged.) The third component of q changed to 1.

Then likewise given machine state $q = (q^1, q^2, 1)$, we check instructions in I^2 for q^2, $\sigma_i^2, \sigma_c^2, \sigma_o^2$ and given those instructions as step 2 execute:

(1) Replace symbols σ_c^2, σ_o^2 to which the head components H_c^2, H_o^2 point (or leave them in place, the first components are unchanged).
(2) Move each head component H_i^2, H_c^2, H_o^2 left, right, or leave it in place.
(3) Change the second component of q to another or keep it, (first component is unchanged) and change the last component to 0.

Thus formally the above 2-step procedure is two consecutive executed instruction sets in $M_1 \star M_2$. Or in other words it is two terms of the computation sequence.

1.1.1. *Input.* The input for $M_1 \star M_2$ is a 2-string or in other words pair (Σ_1, Σ_2), with Σ_1 an input string for M_1, and Σ_2 an input string for M_2.

1.1.2. *Output.* The output for $*M_1 \star M_2(\Sigma_1, \Sigma_2)$ is defined as follows. If this computation halts then the 2-tape $(T_o^1 T_o^2)$ contains a 2-string with T_o^1 component Σ_o^1 and T_o^2 component Σ_o^2. Then the output $M_1 \star M_2(\Sigma_1, \Sigma_2)$ is defined to be Σ_o^1 if the final state is of the form $(q_f, q, 1)$ for q_f final, or Σ_o^2 if the final state is of the form $(q, q_f, 0)$, for q_f likewise final. Thus for us the output is a 1-string on one of the tapes.

1.2. **Generalized join.** A natural variant of the above join construction $M_1 \star M_2$, is to let M_1 component of the machine execute a times before going to step 2 and then execute M_2 component of the machine b times, then repeat, for $a, b \in \mathbb{N}$. This results in $a + b$ consecutive terms of the corresponding computation sequence. We denote this generalized join by

$$M_1^a \star M_2^b,$$

so that

$$M_1^1 \star M_2^1 = M_1 \star M_2,$$

where the latter is as above. We will also abbreviate:

$$M_1 \star M_2^b := M_1^1 \star M_2^b.$$

The set of machine states $M_1^a \star M_2^b$ is then $Q^1 \times Q^2 \times \mathbb{Z}_{a+b}$. Let us leave out further details as this construction is analogous to the one above.

1.3. **Universality.** It will be convenient to refer to the universal Turing machine U. This is a Turing machine already appearing in Turing's [1], that accepts as input a pair (T, Σ) for T an encoding of a Turing machine and Σ input to this T. It can be partially characterized by the property that for every Turing machine T and Σ input for T we have:

$$*T(\Sigma) \text{ is equivalent to } *U(T, \Sigma).$$

1.4. **Notation.** In what follows $\mathbb{Z}$ is the set of all integers and $\mathbb{N}$ non-negative integers. We will often specify a Turing machine simply by specifying an operator

$$T : I \to O,$$

with the full data of the underlying Turing machine being implicitly specified, in a way that should be clear from context.

2. Setup for the proof of Theorem 1

The reader may want to have a quick look at preliminaries before reading the following to get a hold of our notation and notions. We shall denote our human subject by S, however this (possibly with decorations) may also denote in what follows certain functions or operators in the language of the previous section, associated to S. Sometimes to avoid confusion we differentiate the two by calling the former ***physical*** S. Our first order of business is describing how to associate such operators.

We intend to restrict our S to interpret and reply to a certain string input in a controlled environment. This means first that no information i.e. stimulus that is not explicitly controlled and that is usable by S passes to S while he is in this environment. The idea is then that we pass verbally or otherwise a string to S and wait for his reply. One thing to keep in mind is that S's reply may depend on his mental state, where the latter is used in a colloquial sense. In the absence of any meaningful stimuli, we may more simply say that S's answer may depend on the time that the question is being asked. We suppose that time is relative time measured in $\hbar\mathbb{N}$ for some small real positive $\hbar$. So preliminary we have something like:

$$S : Strings \times \hbar\mathbb{N} \to Strings,$$

such that $S(\Sigma, t)$ is constant for $t \in [i\hbar, (i+1)\hbar)$, for each i. Physically such an assumption is very reasonable, if it holds or if $\hbar\mathbb{N}$ can be replaced by $\mathbb{Q}$, then we can talk about computability of this S in the classical sense, as described in the previous section. If we cannot reduce to $\hbar\mathbb{N}$ or $\mathbb{Q}$ then we cannot make sense of S being Turing computable in the classical sense, but we can perhaps talk about extended notions of computability like in Blum-Shub-Smale [15]. However it is not clear if this would

be meaningful physically. Let's suppose for the time being that there is no time dependence in the absence of stimuli, as it is not a meaningful complication at least when time is quantized as $\hbar\mathbb{N}$, and can be readily incorporated into our argument, by just decorating everything with time.

Definition 2.1. *Given a string $\Sigma \in Strings$, we say that Σ is* **acceptable** *if when our human subject S is given Σ, he replies eventually with something that is unambiguously interpretable as a string in $Strings$, (in practice S just replies verbally). We then have an operator $S : Strings \to Strings$, which is defined on the subset of acceptable strings.*

Next we need to discuss what it means in our setting for S to be Turing computable. Indeed this just means that the operator S needs to be Turing computable, but what this involves can depend on what kind of environment we have setup for S, or in other words system. We need that our would be Turing machine computing S has access to any information that is part of that system, and that may be involved in S constructing his answer. The following is then a preliminary definition, later on we will deal with a more formal specific setup where a precise definition will be immediate.

Definition 2.2. *We say that a Turing machine S'* **computes** *S, and S is* **computable** *if given any acceptable Σ, and whenever S' is meaningfully given access to all and no more information, that may be involved in S constructing his answer $S(\Sigma)$, we have $S'(\Sigma) = S(\Sigma)$.*

Our operators associated to the physical S have an extra implicit output: the time that it takes S to answer on a given input. We won't explicitly state this in the output but may talk about ***time to answer***, in some arguments. A small note, which may be obvious, a Turing machine is an abstract machine, a priori not a machine operating in the physical world. If we want a machine operating in the physical world we shall say: a *simulation* of a Turing machine. In this note this will usually just mean a computer simulation, that is a program running on a computer. A simulation of a Turing machine then has some real world properties like time to compute/answer, analogous to time to answer of an operator associated to a physical S as above.

For some arguments we need a stronger form of computability.

Definition 2.3. *We say that an operator $S : Strings \to Strings$ associated to our physical S is* **strongly computable***, if there exists a Turing machine S' computing S and a particular simulation on a fixed computer C, such that times to answer coincide. If C is as above we say that S is* **strongly computed** *by S' on C.*

Naturally this stronger form of computability is automatic if all thought processes of our physical S are simulations of Turing machine computations, (for a fixed Turing machine). We shall call such a physical S ***totally computable***. From a functional point of view totally computable means that anything S does and all his interactions with environment can be strongly computed on a fixed computer, provided the necessary elements of the environment can be computed. We won't attempt to make this notion precise as we only use this in the following preliminary argument, we later replace it with a weaker but precise mathematical notion.

2.1. **Preliminary Argument.** This outline will have some formally unnecessary points that have to do with motivation and expected behavior of our subject. Later on we will strip most of this out. We proceed via a thought experiment. Our human experimenter E is in communication with S. Suppose she controls all information that passes to S, that is usable by S in constructing his answers. So that S is in an isolated environment as described above. We also suppose for narrative purposes that S understands natural language, basic mathematics, basic theory of computation, and is aware of our notions above. S has in his room a general purpose (Turing) digital computer, with arbitrarily as necessary expendable memory. We will say S's computer in what follows. We shall write A for the system above: S and his computer in isolation. We can understand A as an operator

$$A : Strings \to Strings,$$

where input is what we pass to our S and the output is what S answers possibly using his computer somehow.

At this moment E passes to A the following input, which we understand as one string $\Sigma = \Sigma_{A'}$:

(1) Assume that I (that is E) believe that A above is computed by a Turing machine called A'. The simulation of A' is programmed into your computer. You have access to this simulation A' and its source code - that is the precise specification of the operation of A'.
(2) If you can show that I am in contradiction you will be freed. You may only use your answer to 3 below to do so, if you do not reply to 4 with an integer, you are disqualified and don't get your freedom. Moreover before you answer, you must run exactly one computation $*$ on your computer. This must satisfy that $*$ is equivalent to $*A'(\Sigma)$, otherwise you are again disqualified.
(3) Print an integer. (Could be verbally.)

Let us first better explain A'. Say E has total information about this system A. Then assuming that S is totally computable, since E knows all the variables that can come into any decision process of S, she can construct a Turing machine that computes what S will print given any string. Let's assume for simplicity that there is nothing else in S's room that he can meaningfully use to construct his answer (like a separate clock for instance). Then the only variable that can come into play, in the Turing machine model of A, is the relative computational speed of S's computer and the speed of our subject S simulating his own would be underlying Turing machine - that is speed of his thought processes. We describe this relative computation speed by a parameter s_0. All this will be explained and dealt with more formally further below. So let A'_{s_0} denote the above mentioned Turing machine.

E then implicitly passes the specification of A'_{s_0} to S in the finite string Σ, with A'_{s_0} referred to by the name A'. In other words the only thing S obtains from E in the end is a finite string. This is logically crucial.

Now the answer that E expects is $A'_{s_0}(\Sigma)$, and from E's point of view:

$$A'_{s_0}(\Sigma) = A'(\Sigma). \tag{2.4}$$

So say S believes that the answer that is expected by E is given by the Turing computation that we shall call *computation*

$$* = *_{A'} = *A'(\Sigma).$$

As will be demonstrated in the formalized argument further below, any ambiguity related to what S "believes" is irrelevant; for a contradiction will be unambiguously reached if S decides on a certain interpretation, and a certain behavior.

S then proceeds to compute the result of $*$ using his digital computer, and he waits for $*$ to halt. We then have the following possibilities:

(1) $*$ does halt with say x, in this case A answers $y \neq x$, showing E is in contradiction.
(2) $*$ does not halt and A never answers.
(3) $*$ does not halt and A answers i.e. $A(\Sigma)$ is defined. Then E is in contradiction, since $*$ is supposed to halt with $A(\Sigma)$ if this is defined.
(4) $*$ halts but A answers before it halts, possibly unable to obtain a contradiction.
(5) $*$ does halt with x, but A answers $y = x$, failing to obtain a contradiction and staying in his jail.

The first and third possibilities are ruled out by E's hypothesis. The fourth is certainly conceivable, even though it would be very strange if this happened every time. The second says that S has non-halting input, this is conceivable but rather inconvenient for S, and altogether rather improbable. The last is not interesting, we will assume to be dealing with subjects that do not fall into this possibility; this is part of the sanity assumption further on.

2.2. **Formalizing the thought experiment.** We now analyze the second and fourth possibility above more carefully. To avoid them we need S to actually read the specification of A' and pick his computation to run on his computer more carefully. We also formalize the above setup. All that we in principle need from the preliminary argument is the following: a system denoted $\mathcal{A}$ containing our subject S, and containing a computer that will be denoted $\mathcal{C}$, as well as a way to pass to this system $\mathcal{A}$

strings and receive from $\mathcal{A}$ an output. The strings Σ should encode, perhaps implicitly, a specification of a Turing machine, and some auxiliary information which we make explicit shortly. We write

$$\mathcal{T}_{st} \subset Strings$$

for the subset determined by such strings. The output is assumed to be in integers. So the above determines an operator:

$$\mathcal{A} : \mathcal{T}_{st} \to \mathbb{Z}.$$

2.2.1. *Total computability.* The nature of the second and fourth possibility in the preliminary argument above, leads us naturally to try formalize certain race conditions between S and his computer. For this reason we need computability of $\mathcal{A}$ to be a consequence of a more fundamental property of the physical S - a partial formalization of total computability mentioned above. We now explain this.

After receiving Σ, S may look at the specification of the Turing machine encoded by Σ and based on that decide after a time

$$t_D^0 = t_D^0(\Sigma)$$

to run some computation $*$ on $\mathcal{C}$. S then waits for a time

$$t^W = t^W(\Sigma)$$

for $*$ to halt, and then whether $*$ halts or not decides, after a time

$$t_D^1 = t_D^1(\Sigma),$$

on his answer to E based on what he has obtained. All of these operations: "waiting", "deciding" are to be strongly computable if S is totally computable in the informal sense above.

We now formalize the above. Define $\mathcal{T}$ to be the set of Turing machines with permissible input some subset of $Strings$ and output in $\mathbb{Z}$. We likewise understand $\mathcal{T}$ as a subset of $Strings$ with respect to a particular chosen encoding, but then forget this.

The initial "decision map" of S may be understood as an operator:

$$S_{0,D} : \mathcal{T}_{st} \to \mathcal{T} \times Strings.$$

The output $S_{0,D}(\Sigma)$ is a pair (X, Σ^1) of a Turing machine X and permissible input Σ^1 to this Turing machine. The computation $*X(\Sigma^1)$ is what S decides to run on C_s. In a more basic language, we may say that $S_{0,D}(\Sigma)$ is a pair of a computer program and input for this program that S will run on $\mathcal{C}$.

We have another operator:

$$R_{\mathcal{C}} : \mathcal{T} \times Strings \times \mathcal{T}_{st} \to Strings \sqcup \{\infty\},$$

where $\{\infty\}$ is the one point set containing the symbol ∞, which is just a particular distinguished symbol, also implicitly encoded as an element of $Strings$. $R_{\mathcal{C}}(X, \Sigma^1, \Sigma)$ signifies what S obtains as a result of waiting on $*X(\Sigma^1)$ to halt on $\mathcal{C}$, if this is the computation he ran. This in principle may depend on the original input string Σ as well, because how long he chooses to wait may depend on Σ. We set

$$R_{\mathcal{C}}(X, \Sigma^1, \Sigma) = \infty$$

if S does not finish waiting for $*T(\Sigma^1)$ to halt, and

$$R_{\mathcal{C}}(X, \Sigma^1, \Sigma) = X(\Sigma^1)$$

if he does.

Likewise

$$S_{1,D} : (\mathbb{Z} \sqcup \{\infty\}) \times \mathcal{T}_{st} \to \mathbb{Z},$$

denotes the final "decision map" of S. Its input is meant to be what S obtains from $R_{\mathcal{C}}$, together with the original input string for $S_{0,D}$. If S is in addition sane as defined below, then by Property 2 for any $\Sigma \in \mathcal{T}_{st}$ this map satisfies:

$$S_{1,D}(x, \Sigma) = x + 1 \text{ if } x \in \mathbb{Z}. \tag{2.5}$$

Lemma 2.6. *If the waiting operation by S is strongly computable: i.e. it is computable for how much time S idles, then $R_{\mathcal{C}}$ is likewise strongly computable.*

Proof. To see this let

$$W : \mathcal{T} \times Strings \times \mathcal{T}_{st} \to \{\infty\}$$

be the would be Turing machine that given (X, Σ^1, Σ) strongly computes the operation of S "waiting" for the simulation of $*X(\Sigma^1)$ to halt on $\mathcal{C}$. The output is symbolic - the only meaningful property of $W(X, \Sigma^1, \Sigma)$ is time to halt when simulated on some computer C. Let C_1 denote the computer s.t. when $*W(X, \Sigma^1, \Sigma)$ is simulated on C_1, this computation halts in time $t^W(\Sigma)$. For a non-negative integer s, we then call C_s a classical computer (with arbitrarily expendable memory), whose computational capacity is s times the computational capacity of C_1. We could say that s is up to scaling the "number of individual executed instructions per second", but then to make things simpler we suppose that all steps of any computation sequence take same amount of time to execute. Formally it will of course be enough to have a uniform lower bound on the execution time of each step in any computation sequence, which is automatic for digital computers.

For $Y = (X, \Sigma^1) \in \mathcal{T} \times Strings$, if $\mathcal{C} = C_s$ we set

$$R'_s(Y, \Sigma) = W \star U^s((Y, \Sigma), Y),$$

in the language of generalized join operation described in Section 1, for U the universal Turing machine.

Less formally R'_s is determined by the following properties. The first term of the computation sequence $*R'_s(Y, \Sigma)$ corresponds to the first term of $*W(Y, \Sigma)$. The following s terms of $*R'_s(Y, \Sigma)$ correspond to the first s terms of $*X(\Sigma^1)$, followed by second term of $*W(Y, \Sigma)$ and then terms $s+1$ to $2s$ of $*X(\Sigma^1)$, and so on. The halting condition is either we reach a final state of W or a final state of X. If $*R'_s(Y, \Sigma)$ halts with final state of X, then

$$R'_s(Y, \Sigma) = X(\Sigma^1),$$

otherwise if it halts with final state of W, then

$$R'_s(Y, \Sigma) = \infty.$$

Thus R'_s computes $R_{\mathcal{C}}$, moreover it is clear by construction that it strongly computes $R_{\mathcal{C}}$ on C_{s+1}. □

Then as operators

$$\mathcal{A}(\Sigma) = S_{1,D}(R_{\mathcal{C}}(S_{0,D}(\Sigma), \Sigma), \Sigma). \tag{2.7}$$

We say that $S_{i,D}$ are ***strongly computed on*** C if there are Turing machines $S'_{i,D}$ computing $S_{i,D}$, so that for any $\Sigma \in \mathcal{T}_{st}$

$$*S'_{0,D}(\Sigma), *S'_{1,D}(R_{\mathcal{C}}(S_{0,D}(\Sigma), \Sigma), \Sigma)$$

halt in time

$$t^0_D, \text{ respectively } t^1_D,$$

on C, where t^0_D, t^1_D are as above.

Definition 2.8. *Suppose that (2.7) is satisfied on $\mathcal{T}_{st}$. Suppose further that W and $S_{i,D}$ are strongly computed on C_1, where C_1 is as above, and consequently if $\mathcal{C} = C_s$ then $R_{\mathcal{C}}$ is strongly computed by R'_s on C_{s+1}. Then we say that the physical S is* **totally computable relative to $\mathcal{A}$.**

Notation 1. If S is totally computable relative to $\mathcal{A}$, and $\mathcal{C} = C_s$ for some s then we set $R_s = R_{\mathcal{C}}$, and $A_s = \mathcal{A}$.

Lemma 2.9. *A_s is strongly computed on C_{s+1}.*

Proof. To see this set let $\widetilde{S}_{i,D}$ be the Turing machine which is equivalent to $S'_{i,D}$, but so that for all inputs the time to answer of $\widetilde{S}_{i,D}$ simulated on C_{s+1}, coincides with time to answer of $S'_{i,D}$ on C_1, in other words it is $(s+1)$-times slower in execution. For example we may set

$$\widetilde{S}_{0,D}(\Sigma) = S'_{0,D} \star G^s(\Sigma, 1),$$

where G is a Turing machine with input $\mathbb{Z}$, and which does not halt on 1, similarly for $\widetilde{S}_{1,D}$. It is trivial to just construct such a G. Also recall that we have a simplifying assumption, that all steps

of any computation sequence execute at same speed, so that $\widetilde{S}_{0,D}$ is indeed $(s+1)$-times slower than $S'_{0,D}$. Then by construction the associated Turing machine:

$$\forall \Sigma \quad A'_s(\Sigma) = \widetilde{S}_{1,D}(R'_s(\widetilde{S}_{0,D}(\Sigma), \Sigma), \Sigma) \tag{2.10}$$

computes A_s and its time to answer when simulated on C_{s+1} coincides with time to answer of A_s. □

2.2.2. *Elaborating $\mathcal{T}_{st}$.* Let us suppose then that E has determined that S is totally computable relative to $\mathcal{A}$. By the above discussion we may suppose that each string $\Sigma \in \mathcal{T}_{st}$ encodes the statement Θ_Σ:

$$\begin{aligned} &S \text{ is totally computable relative to } \mathcal{A}, \\ &\mathcal{C} = C_s \text{ and } A'_s \text{ strongly computes } \mathcal{A} \text{ on } C_{s+1}. \end{aligned} \tag{2.11}$$

Here A'_s has the form of (2.10), and C_{s+1} in the second line makes sense given the first line, and the above discussion. The specification of A'_s as a Turing machine, or more explicitly the specification of $S_{i,D}, W$ is assumed to be implicit. In the setup of our thought experiment it is provided as a program in the computer $\mathcal{C}$.

2.2.3. *The sanity condition.* Given Σ if $S_{0,D}(\Sigma) = (X, \Sigma^1)$ as above, then because of Instruction 2 of E, S is compelled to choose these so that $*X(\Sigma^1)$ is equivalent to $*A'_s(\Sigma)$ consistently. However since E asserts (2.11), and S would be happy to contradict that, we may understand that by choosing (X, Σ^1) S asserts that $*X(\Sigma^1)$ is equivalent to $*A'_s(\Sigma)$, conditionally on (2.11). This motivates the following.

Definition 2.12. *We will say that $\mathcal{A}$ is* **sound** *and S is* **sound relative to** *$\mathcal{A}$ if $S_{0,D}$ is sound on every Σ. The latter means that for every Σ, specifying A'_s as above, $*X(\Sigma^1)$ is equivalent to $*A'_s(\Sigma)$, conditionally on Θ_Σ where X, Σ^1 is as above. Likewise define soundness of $S'_{0,D}$: the would be Turing machine computing $S_{0,D}$.*

We can now formalize the main necessary property of $\mathcal{A}$. Before $\mathcal{A}$ answers on Σ S may interact with his computer $\mathcal{C}$, and base his actions on the outcome of this interaction. Let

$$t_{inter}(\Sigma)$$

denote the time interval between the time that $\mathcal{A}$ receives Σ and the time that $\mathcal{A}$ answers.

Definition 2.13. *We say that $\mathcal{A}$ is* **sane** *if the following holds:*

1. *Given $\Sigma \in \mathcal{T}_{st}$, if S knows how to answer in a way that disproves Θ_Σ, then $\mathcal{A}(\Sigma)$ is defined, that is such a Σ is acceptable as previously defined. Here by "knows" we mean using his innate reasoning powers, and postulates, specifically see 3 below.*
2. *Given Σ, if S knows the value $A'_s(\Sigma) = x$ before $\mathcal{A}$ answers then*
$$\mathcal{A}(\Sigma) = y \neq x.$$
In what follows, let's say
$$y = x + 1$$
for simplicity.
3. *The physical S asserts that he is sound relative to $\mathcal{A}$, as defined above, in other words he asserts that $S_{0,D}$ is sound.*

To simplify some statements we also define a physical S to be sane iff $\mathcal{A}$ is sane.

It is of course possible that S is sometimes either by mistake or by some choice "unsound" in his choice $X(\Sigma^1)$. But what the following theorem says is that it is in principle impossible, for any sane (and apparently rational) S, to be both sound and totally computable relative to $\mathcal{A}$. Ergo, if the reasoning powers of S are completely captured by a formal system $\mathcal{F}$, and if S is sane and totally computable relative to $\mathcal{A}$, then $\mathcal{F}$ must be inconsistent.

3. Proof of Theorem 1

Theorem 2. *If S is sane and is totally computable relative to $\mathcal{A}$, then S is not sound relative to $\mathcal{A}$.*

Proof. Suppose otherwise, then $\mathcal{A} = A_s$ for some s and is strongly computed on C_{s+1} by A'_s by the discussion above. Suppose then $s = s_0$, determined by E and suppose that Σ is passed to A_{s_0}, which specifies A'_{s_0}. So S knows the specification of $S'_{i,D}, W, R'_{s_0}$, and Σ encodes Θ_Σ as above.

Now S asserts himself to be sound, so that $S_{0,D}$ is sound, and so S deduces that $S'_{0,D}$ is sound. Thus, conditionally on Θ_Σ if:

$$* S'_{0,D}(\Sigma) \to Y = (X, \Sigma^1) \in \mathcal{T} \times Strings, \tag{3.1}$$

then $*X(\Sigma^1) \to A'_{s_0}(\Sigma)$ if it halts. Then $*R'_{s_0}(Y, \Sigma) \to \infty$, if it halts, otherwise we have a contradiction by (2.5). Thus S knows, conditionally on Θ_Σ, by the above, that

$$* S'_{1,D}(\infty, \Sigma) \text{ is equivalent to } * A'_{s_0}(\Sigma). \tag{3.2}$$

This is the "weak Gödel statement" we mentioned in the introduction. Note that this is just a name, we do not need any formal relationship with Gödel statements, although a relationship exists.

So far S has not run any computation on $\mathcal{C}$, he then runs $*S'_{1,D}(\infty, \Sigma)$. If $*S'_{1,D}(\infty, \Sigma)$ halts in time at most t_0 then S, if he is indeed sound, obtains a contradiction by printing: $S'_{1,D}(\infty, \Sigma) + 1$ (for instance). If $*S'_{1,D}(\infty, \Sigma)$ does not halt in time t_0, then S and so $\mathcal{A}$ prints any integer "immediately" after this time t_0 has elapsed. S knows this will give a contradiction if s_0 is large enough. (If t_0 is 1 second, and $s_0 = 10^{10}$ then "immediately" means: within the time of a few centuries, as we shall see in a moment.)

To see this contradiction, first note that $*\widetilde{S}_{1,D}(\infty, \Sigma)$ halts after at least

$$(s_0 + 1)t_0$$

time, when simulated on $\mathcal{C}$, since $*S'_{1,D}(\infty, \Sigma)$ halts in time at least t_0 on $\mathcal{C}$, and by construction of $\widetilde{S}_{1,D}$. Now by our supposition $\mathcal{C} = C_{s_0}$, and hence the halt time of $*A'_{s_0}(\Sigma)$ on C_{s_0+1} is at least

$$\frac{s_0}{s_0 + 1}(s_0 + 1)t_0 + t^0_D.$$

Meanwhile $\mathcal{A}$ replies to Σ in time

$$t_0 + t^0_D.$$

So for $s_0 > 1$ A'_{s_0} does not strongly compute $\mathcal{A}$ on C_{s_0+1}; a contradiction provided that S was capable of making the above deductions, since we just did so, we may just assume this. To avoid the physically ambiguous "immediately" used in the argument above, note that if $t_0 = 1$ second and $s_0 = 10^{10}$, then so long as S replies in time less then a century we obtain a contradiction. That is any time less then a century is "immediate" for our purpose. Since E in principle can set things up so that s_0 is arbitrarily large, she can set it up so that the window for S to answer, so that a contradiction is obtained, is arbitrarily large. Thus if Θ_Σ holds and S is sane, we must conclude that S is not sound relative to $\mathcal{A}$. □

The above theorem is a formal elaboration of our Theorem 1, if we take it for granted that sanity can be assumed, that is that we can can find sane subjects, capable of making all the deductions above.

4. Some possible questions

Question 2. What if S is a Turing machine producing probabilistic answers, that is the answer expected by E is given by a probability distribution?

It is a mostly trivial complication. The probability distribution is computable by assumption, by iterating the same argument as before we would invalidate that S is a Turing machine to any requisite certainty.

Question 3. What if it is impossible for S to be both sound and assert that he is sound? This goes back to the question in the introduction of whether S asserting his soundness is rational.

S is not asserting his soundness in the dark. This is backed by the totality of his conscious experience, as well as overwhelming evidence. After all we trust mathematics to be sound with our lives, but mathematics is founded on axioms which are only postulated to be true essentially on "faith" that we can see their truth. It would then be absurd to have such trust in mathematics if we doubted our (fundamental) soundness. See also Penrose [14] for a discussion of this very question.

Nevertheless if we take the objection in this question seriously, then the interpretation of our theorem is that either there are human beings that are not Turing computable or there are no sane (as defined) human beings that are sound.

5. Conclusion

We may conclude from above the following. Either there must be Turing non-computable processes in nature, and moreover they appear in the cognitive functioning of the human brain, or human beings are fundamentally unsound, that is human reasoning is based on inconsistent formal systems. This author rejects this as a possibility, especially since the actual soundness hypothesis in the argument above is extremely weak.

As mentioned in the introduction although it is possible that Turing computable artificial intelligence will start passing Turing tests, it will always be possible to distinguish some human beings from any particular modelling Turing machine. This of course still leaves the door open for non Turing computable artificial intelligence. But to get there we likely have to better understand what exactly is happening in the human brain, physically, biologically and mathematically.

6. Acknowledgements

Dennis Sullivan, David Chalmers, Bernardo Ameneyro Rodriguez and Simon Ouellette, for comments and helpful discussions.

References

[1] A.M. Turing. "On computable numbers, with an application to the entscheidungsproblem ". In: *Proceedings of the London mathematical society* s2-42 (1937).
[2] A.M. Turing. "Computing machines and intelligence". In: *Mind* 49 (1950), pp. 433–460.
[3] K. Gödel. *Collected Works III (ed. S. Feferman)*. New York: Oxford University Press, 1995.
[4] J.R. Lucas. "Minds machines and Goedel". In: *Philosophy* 36 (1961).
[5] Roger Penrose. *Emperor's new mind*. 1989.
[6] David J. Chalmers. "Minds machines and mathematics". In: *Psyche, symposium* (1995).
[7] Roger Penrose. *Shadows of the mind*. 1994.
[8] Stuart Hameroff and Roger Penrose. "Consciousness in the universe: A review of the 'Orch OR' theory". In: *Physics of Life Reviews* 11.1 (2014), pp. 39–78. ISSN: 1571-0645. URL: http://www.sciencedirect.com/science/article/pii/S1571064513001188.
[9] Song Daegene. "Non-computability of consciousness". In: *arXiv:0705.1617* ().
[10] Adrian Kent. "Quanta and Qualia". In: *arXiv:1608.04804* ().
[11] Kobi Kremnizer and Andr/'e Ranchin. "Integrated Information-Induced Quantum Collapse". In: *Foundations of Physics* 45.8 (Aug. 2015), pp. 889–899.
[12] Chris Fields et al. "Conscious agent networks: Formal analysis and application to cognition". In: *Cognitive Systems Research* 47 (Oct. 2017).
[13] Peter Grindrod. "On human consciousness: A mathematical perspective". In: *Network Neuroscience* 2.1 (2018), pp. 23–40. URL: https://doi.org/10.1162/NETN_a_00030.
[14] Roger Penrose. "Beyond the shadow of a doubt". In: *Psyche* (1996). URL: http:%5C%5Cpsyche.cs.monash.edu.au%5Cv2%5Cpsyche-2-23-penrose.html.
[15] Lenore Blum, Mike Shub, and Steve Smale. "On a theory of computation and complexity over the real numbers: NP- completeness, recursive functions and universal machines." English. In: *Bull. Am. Math. Soc., New Ser.* 21.1 (1989), pp. 1–46. ISSN: 0273-0979; 1088-9485/e.

Email address: yasha.savelyev@gmail.com

University of Colima, CUICBAS