NON-COMPUTABILITY OF HUMAN INTELLIGENCE

YASHA SAVELYEV

Abstract. We revisit the question (most famously) initiated by Turing: can human intelligence be completely modelled by a Turing machine? To give away the ending we show here 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. 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 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 an 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 a bit 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 laws. We would rather not get into detailed discussion what deterministic means here exactly, since from some point of view

quantum mechanics is not deterministic. Suffice to say that for our purpose here, quantum mechanics is 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, [8, 310]. Later Lucas [7] and Penrose [9] 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. For disclosure this author does believe in consistency of human beings, but to me consistency always seemed to be an emergent feature of something deeper, (consciousness perhaps). The main common objections to their argument concern the meta-logic of the argument, see [4] for instance. We shall a few words about this meta-logical issue below, in the context of our version of Lucas-Penrose argument, as well as suggest a fix. 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 for instance [9].

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 [6], on how this can take place in the human brain. Another physical argument for non-computability is presented in Song [5].

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, at least 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 formal systems but none of this appears in the main argument. Let us review the Lucas-Penrose argument, but we reformulate this a bit to better match with our discussion later on.

Let P be a human subject, which we understand as a machine printing statement 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_T 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 mentioned above.

Now P replies: I know I am sound, hence T must be sound, by I can then construct the Gödel statement $G(T)$ for T, which is then a true statement in arithmetic, which indirectly asserts "T cannot print $G(T)$". This alone doesn't quite work however for E replies: 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 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. *Possible fix.* (Outline) We can try to fix the above argument as outlined in the following. Change Σ_T to: "here is a specification of a Turing machine T, print your statement that you assert to be true assuming:

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

P then prints $G(T)$, that is $P(\Sigma_T) = G(T)$ and P asserts this to be conditionally true assuming (0.1), so from his point of view he satisfies E's conditions. Now if T computes P, then $G(T)$ is true if P is in fact sound for then T is sound. But $G(T)$ says: T cannot print $G(T)$. So we reach a contradiction, unless P is not in fact sound. The above outline is still not totally satisfactory, for one thing (0.1) is not a statement in arithmetic, so what does it mean to say P and T are (conditionally) sound? Does the construction of Gödel statement $G(T)$ go through? But we delve no further.

This note can be understood as an elaboration of our "fixed" version of Lucas-Penrose 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 construct Gödel statements, for we find an elementary and explicit analogue, a kind of weak Gödel statement directly, and this suffices for S. So Gödel incompleteness theorem is not used at all. 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 common variation.

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 the set of instructions of M, with s_0 the complete configuration whose internal state is q_0. If the sequence is eventually constant the limiting configuration has internal state one of the final states. When the sequence is eventually constant 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 later, 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 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 value, or both do not halt.*

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 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 $Strings^n$, with i'th component 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 $(q^1, q^2, 0)$ 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, 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 q^1 component to another state or keep it. (Likewise the q^2 component is unchanged.) The third component is changed to 1.

Then likewise given machine state $(q^1, q^2, 1)$, 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, 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, (independently).
(3) Change q^2 component to another state or keep it, 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. **Morphisms.** Given a Turing machine M we denote by $\mathcal{S}(M)$ the set of all possible complete configurations of M, (for any possible input). For $s \in \mathcal{S}(M)$ denote by $q(s)$ its internal state.

Definition 1.6. *Given Turing machines*

$$M_1 : \mathcal{I}_1 \to \mathcal{J}_1,$$

$$M_2 : \mathcal{I}_2 \to \mathcal{J}_2$$

an **embedding**

$$m : M_1 \to M_2$$

consists of injective functions:

$$m_i : \mathcal{I}_1 \to \mathcal{I}_2,$$

$$m_o : \mathcal{J}_1 \to \mathcal{J}_2,$$

$$m_{\mathcal{S}} : \mathcal{S}(M_1) \to \mathcal{S}(M_2)$$

satisfying the following properties:

$$q(m(s_1)) = q(m(s_2)),$$

if

$$q(s_1) = q(s_2),$$

and m takes initial state to initial and final states to final. We also require injectivity on machine states:

$$\text{if } q(m(s_1)) = q(m(s_2)) \text{ then } q(s_1)) = q(s_2).$$

Furthermore:

$$\{s_i\}_0^\infty = *M_1(\Sigma)$$

iff

$$\{m(s_i)\}_0^\infty = *M_2(m_i(\Sigma)).$$

Finally we have

$$m_o(M_1(\Sigma)) = M_2(m_i(\Sigma)).$$

This allows us to compute the sequence $*M_1(\Sigma)$ by computing the sequence $*M_2(m_i(\Sigma))$ and vice versa. In this note m_o will always be identity.

1.4. **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 is characterized by the property that for every T there is a natural embedding $e : T \to U$, with

$$e_i(\Sigma) = (T, \Sigma), e_o = id$$

and such that:

$$T(\Sigma) = U(T, \Sigma),$$

for every input string Σ for T.

1.5. **Notation.** In what follows $\mathbb{Z}$ is the set of all integers and $\mathbb{N}$ non-negative integers.

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 depends 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 depends on 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 [3]. 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 $S : Strings \to Strings$ corresponding to S 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.

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 used by S to give his answer $S(\Sigma)$, we have $S'(\Sigma) = S(\Sigma)$.*

Our operators associated to the physical S have an extra hidden output: time to answer. 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. Usually this just means 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.

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' and a particular simulation on a fixed computer C, such that the associated operators coincide, that is $S = S'$ and times to answer coincide. If C is as above we say that S* **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 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 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 $\Sigma = \Sigma_{s_0,A'}$):

(1) Assume that I (that is E) believe that A above is strongly computed by a Turing machine called A'. The simulation of A' is programmed into your computer, which has processing speed s_0 relative to you (s_0 here is a parameter to be specified by E, we explain below). You have access to this simulation A' and its source code - that is the precise specification of the operation of A'.
(2) Assume also that I believe that $A : Strings \to Strings$ is defined on this string Σ.
(3) If you can show that I am in contradiction you will be freed. You may only use your answer to 4 below to do so, moreover before yours answer, you must run exactly one computation $*$ on your computer. This must satisfy that $*$ is equivalent to $*A'(\Sigma)$, (so that if E can determine otherwise S is disqualified and stays in his jail).
(4) Print an integer. (Can of course be verbally, we use the word print because it is suggestive of finality.)

Let us explain the parameter s_0. Say E has total information about this system A. Then assuming that S is totally computable, since she knows all the variables that can come into any (simulated) computation by S, she can construct a Turing machine that computes what S will print given any string. If we 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); [2] 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 underlying Turing machine (that is speed of his thought processes), we describe this by a parameter s_0. All this will be explained and dealt with more formally further below. So let E_{s_0} denote the above mentioned Turing machine.

She then passes this s_0 to S, in the string $\Sigma_{s_0,A'}$. Note that the finite specification of the Turing machine A' is implicitly part of this string. 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 $E_{s_0}(\Sigma_{s_0,A'})$. We suppose that E setup A' so that it properly interprets the parameter s_0 in the string $\Sigma_{s_0,A'}$, i.e. so that:

$$E_{s_0}(\Sigma_{s_0,A'}) = A'(\Sigma_{s_0,A'}). \tag{2.4}$$

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

$$* = *_{s_0,A'} = *A'(\Sigma_{s_0,A'}).$$

[2] His computer clock would be ok.

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 may then proceed to compute the result of $*$ using his digital computer. Now assuming the above and 1, 2 in particular, S knows that $*$ must halt or rather that E must believe that it halts, for $*$ computes $A(\Sigma)$ and 2 says this is defined. So 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, E is still in contradiction, since she supposes that A is everywhere defined and $*$ computes $A(\Sigma)$.
(3) $*$ does not halt and A answers i.e. $A(\Sigma)$ is defined. 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 triple of possibilities are ruled out by E's hypothesis. The fourth is certainly conceivable. The last is not interesting, we will assume to be dealing with subjects that do not fall into this possibility.

2.2. **Formalizing the thought experiment.** We now analyze the fourth possibility above more carefully. We also formalize the above setup. All that we in principle need from the preliminary argument is the following. A system $A = A_s$ containing our subject S and a computer that will be denoted C_s, with s a parameter analogous to s_0 above, and to be explained below. As well as a way to pass to this system strings and receive from A an output. Note that the parameter s is now freely varying, so we really must understand the operator associated to this system as:

$$A : Strings \times \mathbb{N} \to \mathbb{Z}.$$

We write A_s for the restriction $A|_{Strings \times \{s\}}$. We also write

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

for the subset determined by strings $\Sigma_{s',T}$ as above. (Further on we slightly modify this set.) Note that S does not a priori know what is s, from his point of view it is just a system

$$\mathcal{A}, \tag{2.5}$$

while from the point of view of E it is A_s.

Let's call C_s a classical computer (with arbitrarily expendable memory), whose computational capacity is represented by a non-negative integer s, via some mapping so that $C_1 = C$, where the latter is made explicit below (2.9), and where C_s has 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 upper and lower bound on the execution time of each step in any computation sequence, which is automatic for digital computers.

Before A_s answers, S may interact with his computer C_s, and base his actions on the outcome of this interaction. Let

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

denote the time interval between the time that A receives Σ and the time that A answers. We make some basic assumptions about this interaction in what follows.

Definition 2.6. *We say that A is* **sane** *if:*

(1) *If given $\Sigma = \Sigma_{s',T}$, S knows how to answer in a way that unambiguously proves that A_s is not strongly computed by $T_{s'}$ then $A_s(\Sigma)$ is defined, that is such a Σ is acceptable as previously defined.*

(2) *There is a $t_0 = t_0(S) > 0$, so that for all s, if some computation $*$ started by S in A_s, at a moment of time in $t_{inter}(\Sigma)$, halts on C_s in time less then t_0, then S knows the limit of $*$ before $A_s(\Sigma)$ answers. (This just means that S has some minimal patience).*

(3) *Given any s', T, if S knows the value $T_{s'}(\Sigma_{s',T}) = x$ before A_s answers then*

$$A_s(\Sigma_{s',T}) = y \neq x.$$

In what follows, let's say

$$y = x + 1$$

for simplicity.

(4) *The physical S believes himself to be sound, or conditionally sound if he is working on a given fixed assumption. (We ellaborate on what this means further on.)*

To simplify some statements we also define a physical S to be sane iff A is sane.

We need computability of A to be a consequence of a more fundamental property of the physical S - a partial formalization of total computability mentioned above. After receiving Σ, S looks (or possibly does not) at the specification of T and based on that decides after a time t^0_D to run some computation $*$ on C_s. S then waits for a time t^W for $*$ to halt, and then whether $*$ halts or not decides, after a time t^1_D, 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.

We have another operator:

$$R_s : \mathcal{T} \times Strings \to Strings \sqcup \{\infty\},$$

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*. $R_s(T, \Sigma^1)$ signifies what S obtains as a result of waiting on $*T(\Sigma^1)$ to halt on C_s, (if this is the computation he ran.) We set $R_s(T, \Sigma^1) = \infty$ if S does not finish waiting for $*T(\Sigma^1)$ to halt, and $R_s(T, \Sigma^1) = T(\Sigma^1)$ 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_s and the original input string for $S_{0,D}$. By Property 3 of sanity for any $\Sigma \in \mathcal{T}_{st}$ this map satisfies:

$$S_{1,D}(x, \Sigma) = x + 1 \text{ if } x \in \mathbb{Z}. \tag{2.7}$$

If the waiting operation by S is computable: it is computable for how much time S idles, then so is R_s. To see this let

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

be the would be Turing machine that given (X, Σ^1) strongly computes the operation of S "waiting" on $*X(\Sigma^1)$. The output is symbolic - the only meaningful property of $W(X, \Sigma)$ is time to halt when simulated on some C. Next we set

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

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. For $Y = (X, \Sigma^1) \in \mathcal{T} \times Strings$ the first term of $*R'_s(Y)$ corresponds to the first term of $*W(Y)$. The following s terms of $*R'_s(X, \Sigma^1)$ correspond to the first s terms of $*X(\Sigma^1)$, followed by second term of $*W(Y)$ 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 $*R'_s(Y)$ halts with final state of X, then

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

otherwise if it halts with final state of W, then

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

Thus R'_s computes R_s.

Then as operators

$$A_s(\Sigma) = S_{1,D}(R_s(S_{0,D}(\Sigma), \Sigma). \tag{2.8}$$

We say $S_{i,D}$ are ***strongly computable*** if there are Turing machines $S'_{i,D}$, so that for any $\Sigma \in \mathcal{T}_{st}$ $*S_{0,D}(\Sigma)$, $*S_{1,D}(R_s(S_{0,D}(\Sigma)), \Sigma)$ halt in time t^W, respectively $t^0 = t^0_D(\Sigma), t^1_D = t^1(\Sigma)$ on

$$C = C_1, \tag{2.9}$$

where t^W, t^0, t^1 are as above. If R_s is computed by R'_s as above we say it is ***strongly computable*** if for any $\Sigma \in \mathcal{T}_{st}$ $*W(Y_\Sigma)$ halts in time t^W on C, for W as above.

Definition 2.10. *Suppose that (2.8) is satisfied on $\mathcal{T}_{st}$, where equality is equality of operators. Suppose further that R_s is computed by R'_s as above, and that R_s and $S_{i,D}$ are strongly computable. Then we say that the physical S is* **totally computable relative to** *A.*

Note that if S is totally computable relative to A, and C is as above, then A_s is strongly computed on C_{s+1}. To see this set let $\widetilde{S}_{i,D}$ be the Turing machine which is equivalent to $S'_{i,D}$ but so that time to answer of $\widetilde{S}_{i,D}$ simulated C_{s+1} coincides with time to answer of $S'_{i,D}$ on C, 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. (It is easy to just construct one.) And similarly with $\widetilde{S}_{1,D}$. Then by construction the associated Turing machine

$$A'_s(\Sigma) = \widetilde{S}_{1,D}(R'_s(\widetilde{S}_{0,D}(\Sigma), \Sigma). \tag{2.11}$$

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

2.3. **Modified $\mathcal{T}_{st}$.** Let us suppose then that E believes that S is totally computable relative to A. By the above we may then suppose that each string $\Sigma = \Sigma_{s',T} \in \mathcal{T}_{st}$ specifies a formal assertion Θ_Σ:

$$T_{s'} \text{ strongly computes } \mathcal{A} \text{ on } C_{s'+1}, \tag{2.12}$$

see (2.8). Given $\Sigma_{s',T}$ if $S_{0,D}(\Sigma_{s',T}) = (X, \Sigma^1)$ as above, then because of instruction 3 of E, S is compelled to choose these so that $*X(\Sigma^1)$ is equivalent to $*T_{s'}(\Sigma_{s',T})$ consistently. However since E asserts (2.12), and he would be happy to contradict that, we may understand that by choosing (X, Σ^1) S asserts that $*X(\Sigma^1)$ is equivalent to $*T_{s'}(\Sigma_{s',T})$, conditionally on (2.12). This motivates the following.

Definition 2.13. *We will say that A is* **sound** *and S is* **sound relative to** *A if $S_{0,D}$ is sound on every Σ, meaning that for every $\Sigma = \Sigma_{s',T}$ $*X(\Sigma^1)$ is equivalent to $*T_s(\Sigma_{s',T})$, conditionally on Θ_Σ where X, Σ^1 is as above. Likewise define soundness for A' and $S'_{0,D}$: the would be Turing machines computing $A, S_{0,D}$.*

Remark 2. Naturally, it is possible that S is sometimes either by mistake or by some choice "unsound" in his choice of $X(\Sigma^1)$. But what the following theorem shows is that it is in *principle* impossible for some apparently rational S to be both sound on $\Sigma_{s',A'}$, and totally computable relative to A by A'.

To expand the axiom 4 of sanity, this means that S believes that $S_{0,D}$ is sound as above.

3. Proof of Theorem 1

Theorem 2. *If S is sane and is totally computable relative to A, then S is not sound relative to A (for A as above.)*

Proof. Suppose otherwise, then A_s is strongly computed on C_{s+1} by A'_s by the discussion above. Suppose then that $\Sigma = \Sigma_{s,A'}$ is passed to A_s. So S knows the specification of $S'_{i,D}, R'_s$. And Σ specifies the statement:

$$A'_s \text{ strongly computes } A_s \text{ on } C_s. \tag{3.1}$$

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

$$*S'_{0,D}(\Sigma) \to Y = (X, \Sigma^1) \in \mathcal{T} \times Strings, \tag{3.2}$$

then $*X(\Sigma^1) \mapsto A'_s(\Sigma)$ if it halts. Then $*R'_s(Y) \to \infty$, if it halts, otherwise we have a contradiction by (2.7). Thus S knows (conditionally, by the above) that

$$*S'_{1,D}(\infty, \Sigma) \text{ is equivalent to } *A'_s(\Sigma). \tag{3.3}$$

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 C_s, 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 A_s prints any integer immediately after this time t_0 elapsed. S knows this will give a contradiction if s is large enough. To see this first note that $*\widetilde{S}_{1,D}(\infty, \Sigma)$ halts after at least

$$(s+1)t_0$$

time, when simulated on C_s, since $*S'_{1,D}(\infty, \Sigma)$ halts in time at least t_0 on C_s. (Recall we have a simplifying assumption that all steps of any computation sequence execute at same speed.) Hence the halt time of $*A'_s(\Sigma)$ on C_{s+1} is at least

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

Meanwhile A_s answers on Σ in time

$$t_0 + t^0_D,$$

which is a contradiction to (3.1). So A' does not compute A; a contradiction provided that S was capable of making the above deductions, since we just did so, we may just assume this. Thus if (3.1) holds, we must conclude that S is not sound relative to 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. Does the same argument not also prove that S is not even deterministic?

By a deterministic machine, in non-specific terms, we mean here a machine that accepts certain inputs and *not necessarily computably* deterministically produces outputs. (Or deterministically produces a probability distribution for the output, recall that *we* also call this deterministic). Then the answer is no. If we try to run the same argument but with Turing machine replaced by a deterministic machine as above, we shall arrive at the following point. Suppose E has detailed information about S's working as a deterministic machine. Suppose she somehow passed all this information to S, in the form of some finite data. But even if she is able to determine S's supposed output $S(\Sigma)$, S may not be able to determine it himself, as it is no longer a matter of a computation. (Nor is it certain she can determine it herself.) Thus S cannot contradict E's expectation since he is unable to determine what she expects, even though he can certainly answer using his natural faculties! Just to give a colorful

example, the answer she expects from S may depend on whether some particular 4 manifolds are diffeomorphic. The latter is a computationally unsolvable problem. There are a lot of other obstructions to running our argument in this case, for instance an important point for us was being able to speed up execution of a certain computation sequence, this no longer makes sense.

Question 3. 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.

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. We reject 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.

6. Acknowledgements

Dennis Sullivan, Bernardo Ameneyro Rodriguez, Simon Ouellette, and Dusa McDuff, for comments and helpful discussions.

References

[1] A.M. Turing, *On computable numbers, with an application to the entscheidungsproblem* , Proceedings of the London mathematical society, s2-42 (1937).
[2] ———, *Computing machines and intelligence*, Mind, 49 (1950), pp. 433–460.
[3] L. Blum, M. Shub, and S. Smale, *On a theory of computation and complexity over the real numbers: NP- completeness, recursive functions and universal machines.*, Bull. Am. Math. Soc., New Ser., 21 (1989), pp. 1–46.
[4] S. Bringsjord and H. Xiao, *A refutation of Penrose's Gödelian case against artificial intelligence*, Journal of Experimental & Theoretical Artificial Intelligence, 12 (2000), pp. 307–329.
[5] S. Daegene, *Non-computability of consciousness*, arXiv:0705.1617.
[6] S. Hameroff and R. Penrose, *Consciousness in the universe: A review of the orch or theory*, Physics of Life Reviews, 11 (2014), pp. 39 – 78.
[7] J.R. Lucas, *Minds machines and Goedel*, Philosophy, 36 (1961).
[8] K. Gödel, *Collected Works III (ed. S. Feferman)*, (1995).
[9] R. Penrose, *Emperor's new mind*, (1989).

E-mail address: yasha.savelyev@gmail.com

University of Colima, CUICBAS