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 either the answer is *no*, or human reasoning is fundamentally unsound. In particular, under the soundness hypothesis, we show that at least some meaningful thought processes of the brain cannot be Turing computable. And so some physical processes are absolutely not Turing computable, in any theoretical physical model, which is not entirely expected. The main idea is to resolve certain meta-logical issues with the well known Lucas-Penrose argument by constructing an analogue of Gödel statement directly, rather than appealing to Gödel incompleteness theorem. The main argument is entirely based on the language of Turing machines, which is relatively elementary and should be generally accessible.

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

An informal definition of a Turing machine (see [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 informally outlined a possible obstruction to a yes answer coming from Gödel's incompleteness theorem. For the incompleteness theorem to play a role we need some assumption on the fundamental soundness of human reasoning. 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. At the moment it should be understood as follows: we make inconsistent assertions not because these are expressions of fundamental internal inconsistencies of our mental constructions, but because the noisy, faulty biological nature of our brain leads to interpretation errors of these mental constructions. Here by faulty, we mean the possibly common occurrence of faults in brain processes, coming from things like cell death, signaling noise between neurons, etc., even if the brain may have robust fault protections built in.

Gödel himself first argued for a no answer in [3, p. 310]. Later Lucas [4] and later again and more robustly Penrose [5] argued for a no answer.

They further formalized and elaborated the obstruction coming from Gödel's incompleteness theorem. And they reject the possibility that humans could be unsound on a fundamental level, as does Gödel but for him it is apparently not even a possibility, it does not seem to be stated in [3]. [1]

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. Here is a partial list of some partially related work on mathematical models of brain activity and or quantum collapse models: [7], [8], [9], [10].

The following is a slightly informal version of our main Theorem 3.1.

[1]It is likely most mathematicians would sympathize with Gödel, after all mathematics is meaningless if mathematicians are fundamentally unsound - for then they must eventually assert everything to be true.

Theorem 0.1. *Either there are meaningful, non Turing computable processes in the human brain, or human beings are fundamentally unsound, in fact the actual soundness condition needed is very weak. Moreover, in case of the former, some specific real world (intelligent) behavior of a fixed human being cannot be simulated by a fixed Turing machine.*

In a sense the above theorem is nothing more then confirmation of the original intuition of Turing, except our argument is different from what he appeared to intend. Broadly interpreted this is a theorem in computer science. But the immediate implications and context are in mathematical physics and in part biology, and philosophy. For even existence of non Turing computable processes in nature is not known. For example we expect beyond reasonable doubt that solutions of Navier-Stokes equations or N-body problems are generally non Turing computable, (over $\mathbb{Z}$, if not over $\mathbb{R}$ cf. [11]), as modeled in essentially classical mechanics. But in a more physically accurate and fundamental model these may both become computable, (possibly if the nature of the universe is ultimately discreet.) Our theorem says that either there are absolutely, that is model independent, non-computable processes in physics, in fact in the functioning of the brain, or human beings are fundamentally unsound, which is a mathematical condition on the functioning of the human brain. Despite the partly physical context the technical methods of the paper are mainly of mathematics and computer science, as we need very few physical assumptions.

Outline of the main idea of the Gödelian attack. What follows will be very close in essence to the argument Penrose gives in [12], which we take to be his main and final argument, (as far as I am aware). However we partially reinterpret to be closer to our argument later on. While this outline uses some of the language of formal systems, except for some supplementary remarks, we will *not* use it in our actual argument, which is based purely on the language of Turing machines, and is more elementary.

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 in a fixed alphabet, $P(\Sigma)$ 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 a specification of a Turing machine T, print your statement that you assert to be true assuming Θ_T:

$$T \textit{ computes } P. \tag{0.2}$$

Before we proceed, we put the condition on our P that he asserts himself to be fundamentally sound.

Now P reasons that since he is fundamentally sound, so that his deductions based on Θ_T are sound, the same must be true of T. On the other hand, as is well known the statements $T(\Sigma)$, must be provable statements in a certain formal system $\mathcal{F}(T)$ associated to T. Then (perhaps) there is a conditional Gödel statement $G(T, \Theta_T)$, which is true if $\mathcal{F}(T) + \Theta_T$ is consistent and indirectly asserts that T cannot print $G(T, \Theta_T)$. Here $\mathcal{F}(T) + \Theta_T$ denotes $\mathcal{F}(T)$ 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 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/formalized, otherwise it is not clear how the interpretation of Θ_T as a statement could work, but we delve no further. A full critique of the Penrose argument, is given in Koellner [13], [14], see also Penrose [15], and Chalmers [16] for discussions on related issues.

This note can be understood as a non-literal elaboration of the above argument, but with additional changes and improvements. First we do not need our subject of this paper, henceforth denoted by 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 directly find in this context an elementary and explicit analogue, a kind of weak Gödel statement and this suffices for S. Thus Gödel incompleteness theorem is not used at all.

However, to construct our "weak Gödel statement" directly, we still need to formalize at least partly some necessary properties of S. To this end we need to work with a slightly stronger form of Turing computability of S, although physically it is only trivially stronger. Roughly speaking for us computability means that not only are the answers of S computable by a Turing machine T, but also that the brain of S is physically simulating this T to arrive at the answers. (The actual condition we use is more elementary and explicit.)

So we solve the problem above, in the "fixed argument", of formalization of (0.2), by avoiding direct use of Gödel incompleteness theorem, and by defining/formalizing our subject S just enough so that a direct construction of the "weak Gödel statement" goes through. This requires a slightly stronger computability condition. In particular we are not immediately at odds with analysis in [13], [14] of the argument of Penrose.

Working with Turing machines, has at least one advantage of being technically more elementary and concrete compared to working with formal systems. In particular the above mentioned weak Gödel statement is formulated purely in the language of Turing machines - as a property of a certain hypothetical Turing machine.

As a final remark, technically the paper is mostly elementary and should be widely readable, in entirety.

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***. 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.*

In practice 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.

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 1.5. 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 some 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 1.6. *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.** Our Turing machine of Definition 1.1 is a multi-tape enhancement 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 $T^1, \ldots, T^n$ in parallel, which we denote by $(T^1 \ldots T^n)$ 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 $(T^1 \ldots T^n)$ is an n-string, formally just an element $\Sigma \in \mathit{Strings}^n$, with i'th component of Σ specifying a string of symbols on T^i. The blank symbol b is the symbol $(b^1, \ldots, b^n)$ with b^i blank symbols of 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 $1, 2$ 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, $j = 1, 2$. It has 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 $(\sigma_i^1 \sigma_i^2), (\sigma_c^1 \sigma_c^2), (\sigma_o^1 \sigma_o^2)$ to which the heads H_i, H_c, H_o are currently pointing, 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, bounded by b symbols, 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 0.1

Our subject is a human called S, also sometimes referred to by ***physical*** S to avoid confusion with similarly named operators. We intend to restrict our S to interpret and reply to certain string input, while in a isolated 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. If time is relative time measured in $\mathbb{R}_{\geq 0}$, then we may associate to S an operator:

$$S : Strings \times \mathbb{R}_{\geq 0} \to Strings,$$

$\mathbb{R}_{\geq 0}$ here is an idealization, physically it must likely be replaced by $\hbar\mathbb{N} \subset \mathbb{R}_{\geq 0}$ for some $\hbar$ small, and without this reduction we cannot even mathematically talk about computability of S, at least over $\mathbb{Z}$, cf. [11]. However we are going to ignore this possible time dependence, as it is not a meaningful complication, 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 whenever 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.*

The following is a preliminary definition, later on we will deal with a more formal specific setup where a precise definition will be immediate.

Definition 2.2. *For physical S in an isolated environment as discussed above, we say that a Turing machine S'* **computes** *S, and S (the operator) is* **computable**, *if for any acceptable Σ 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 the corresponding 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 physical simulations of Turing machine computations, (for a fixed Turing machine). We shall call such a physical S ***totally computable***, we later give a weaker but precise mathematical formulation of this, that will suffice.

2.1. **Preliminary Argument.** We proceed via a thought experiment. Our human experimenter E is in communication with S, who is in an isolated environment as described above. S has in his room a general purpose (Turing) digital computer, with arbitrarily as necessary expendable memory. A will denote the above system: S and his computer in isolation. A is then 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.

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

(1) After receiving all the following instructions print an integer Σ_{out}. (Could be verbally.)
(2) A simulation of a Turing machine called 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'.
(3) Before you answer with Σ_{out} you must run exactly one computation

$$* \tag{2.4}$$

on your computer. If Σ_{out} is not an integer, or is undefined you are disqualified, and you stay imprisoned.

(4) If you can disprove via Σ_{out} the statement Θ_Σ:

$$A' \text{ computes } A,$$

you will be freed. There is a restriction that the following statement O_Σ must hold:

$$\text{If } \Theta_\Sigma \text{ then } * \text{ is equivalent to } *A'(\Sigma),$$

for $*$ as in (2.4). If I (that is E) can determine otherwise you are again disqualified.

Suppose that S proceeds to compute the result of

$$* = *A'(\Sigma)$$

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 $y \neq x$, disproving Θ_Σ.
(2) $*$ is non-halting and A never answers.
(3) $*$ is non-halting and A answers i.e. $A(\Sigma)$ is defined, in this case Θ_Σ is disproved in principle.
(4) $*$ halts but A answers before it halts, in this case S ran out of patience and is possibly unable to obtain a contradiction.
(5) $*$ does halt with x, but A answers $y = x$, failing to disprove Θ_Σ and staying in his jail.

The fourth is certainly conceivable, even though it would be very strange if this happened every time. The second says that *the human* S has non-halting input, this is conceivable but rather inconvenient for S, and altogether rather improbable, especially if S does not read the specification of A'. 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.** To deal with the second and fourth possibility S needs to actually read the specification of A' and pick the computation to run on his computer more carefully, we now explain how this is possible. $\mathcal{A}$ will denote the system containing our subject S, and containing a computer that will be denoted $\mathcal{C}$. Our input strings Σ should encode a specification of a Turing machine. (In the preliminary argument this was implicit.) We write

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

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

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

2.2.1. *Total computability.* As we mentioned S will have to study the specification of A', but to truly be able to make use of this, 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

$$t^0_D = t^0_D(\Sigma)$$

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

$$t^1_D = t^1_D(\Sigma),$$

on his printed 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 discussed. We now further formalize this.

Define $\mathcal{T}$ to be the set of Turing machines with permissible input some subset of *Strings* and output in $\mathbb{Z}$. 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 $\mathcal{C}$. 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 $*X(\Sigma^1)$ 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 4 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

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

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 slightly simpler, we suppose that all steps, of any computation sequence run on C_s, 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, run on a fixed computer, which is automatic for digital computers.

For

$$Y = (X, \Sigma^1) \in \mathcal{T} \times \mathit{Strings}$$

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}. □

We then have an equality of operators:

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

Definition 2.8. *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

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

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

Definition 2.9. *Suppose that (2.7) is satisfied for every $\Sigma \in \mathcal{T}_{st}$, and that W strongly computes the "waiting operation" of S on C_1, as in the proof of Lemma 2.6. And suppose further that $S_{i,D}$ are strongly computed on C_1 by $S'_{i,D}$. 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 2.10. *A_s is strongly computed on C_{s+1}.*

Proof. Let $S'_{i,D}$ be as in the Definition 2.9, and set $\widetilde{S}^{s+1}_{i,D}$ be the Turing machine which is equivalent to $S'_{i,D}$, but so that for all inputs the time to answer of $\widetilde{S}^{s+1}_{i,D}$ 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

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

where G is a Turing machine with input $\mathbb{Z}$, and which does not halt on 1, similarly for $\widetilde{S}^{s+1}_{1,D}$. 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 $\widetilde{S}^{s+1}_{0,D}$ is indeed $(s+1)$-times slower than $S'_{0,D}$.

Then by construction the associated Turing machine:

$$\forall \Sigma \quad A'_s(\Sigma) = \widetilde{S}^{s+1}_{1,D}(R'_s(\widetilde{S}^{s+1}_{0,D}(\Sigma), \Sigma), \Sigma) \tag{2.11}$$

computes A_s and its time to answer when simulated on C_{s+1} coincides with time to answer of A_s. Here R'_s strongly computes R_s on C_{s+1} as in the proof of Lemma 2.6. □

2.2.2. *Elaborating the computability condition.* By the discussion above, the statement which S will use as a conditional can be assumed to be Θ_Σ:

$$\begin{aligned} &S \text{ is totally computable relative to } \mathcal{A}, \\ &\mathcal{C} = C_s \text{ and } A'_s \text{ strongly computes } \mathcal{A} \text{ on } C_{s+1}. \end{aligned} \tag{2.12}$$

Here A'_s has the form of (2.11), 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, as well as $S'_{i,D}, W$ is provided by Σ. In our preliminary argument this was provided as a program in the computer $\mathcal{C}$.

2.2.3. *The soundness and sanity condition.* Given Σ if

$$S_{0,D}(\Sigma) = (X, \Sigma^1)$$

as above, then because of Instruction 4 of E, S is compelled to choose these so that the following statement O_Σ holds:

$$\text{If } \Theta_\Sigma \text{ then } *X(\Sigma^1) \text{ is equivalent to } *A'_s(\Sigma).$$

In other words we may suppose that the choice of (X, Σ^1) is such that S can assert O_Σ. This would be the sane behavior for S, and is part of the more involved "sanity" condition below.

Definition 2.13. *We will say that S is* **sound relative to $\mathcal{A}$**, *if O_Σ holds for every $\Sigma \in \mathcal{T}_{st}$. Likewise define soundness of $S'_{0,D}$, which is as in Definition 2.8. The statement which expresses soundness of S relative to $\mathcal{A}$ will be denoted by $Sound(S)$, while the statement that $S'_{0,D}$ is sound will be denoted by $Sound(S')$.*

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 open time interval between the time that $\mathcal{A}$ receives Σ and the time that $\mathcal{A}$ answers. We now axiomatize the expected properties of our S. We may suppose that these are properties that are partially compelled from S by E.

Definition 2.14. *We say that S is* **sane relative to** $\mathcal{A}$*, or just* **sane***, or* $sane(S)$ *holds, if the following holds:*

(1) *$\mathcal{A}(\Sigma)$ is an integer if defined.*
(2) *S runs a single computation $*$ on his computer in the time interval $t_{inter}(\Sigma)$. In the notation above $* = *X(\Sigma^1)$.*
(3) *Given $\Sigma \in \mathcal{T}_{st}$, if S deduces in the time interval $t_{inter}(\Sigma)$ an integer x, so that $\mathcal{A}(\Sigma) = x$ disproves the conjunction*

$$\Theta_\Sigma \wedge Sound(S') \tag{2.15}$$

then $\mathcal{A}(\Sigma)$ is defined with such an x. In particular such a Σ is acceptable as previously defined. Here by "knows" we mean using his innate reasoning powers and postulates, in particular this can involve 5 below.

(4) *Given Σ if S knows the value $A'_s(\Sigma) = x$ in the time interval $t_{inter}(\Sigma)$ then*

$$\mathcal{A}(\Sigma) = y \neq x.$$

In what follows let's say

$$y = x + 1,$$

for simplicity.

(5) *S asserts $Sound(S)$.*

To elaborate, part of the condition 5 is that S does not knowingly choose his $* = *X(\Sigma^1)$ in contradiction with O_Σ. For example if $* = *T(\Sigma)$, for T any Turing machine computing the operator

$$\Sigma \mapsto A'_s(\Sigma) + 3,$$

then for every Σ:

$$*X(\Sigma^1) \text{ is } \textit{not} \text{ equivalent to } *A'_s(\Sigma).$$

So this choice of $*$ would certainly prove that $\neg Sound(S')$ if Θ_Σ, but it would disqualify S by the instructions of E, since $\neg O_\Sigma$ and E requires O_Σ. The sanity condition of S states that this is something that he asserts to have avoided. So the goal of S is not to just disprove the conjunction in (2.15), which as we see is trivial, but to do so while being bounded by the sanity condition.

The reason we need S bounded by the sanity condition is that in this case if we demonstrated $\neg Sound(S)$, then we could conclude that S is not fundamentally sound, since S asserts $Sound(S)$ in absolute faith, see however Question 3 below.

If we assume $sane(S)$, then the string Σ needs to be purely a specification of a Turing machine. All the other instructions encoded in the string Σ of the preliminary argument had to do with compelling the "sane", as above, behavior from S, and are formally unnecessary, once $sane(S)$ is assumed.

3. Proof of Theorem 0.1

Theorem 3.1. *If $sane(S)$ then for every Σ either not Θ_Σ or not $Sound(S)$, and in the latter case S is fundamentally unsound.*

Proof. Suppose that $\Sigma_0 \in \mathcal{T}_{st}$ is passed to $\mathcal{A}$. Then Σ_0 specifies the Turing machine A'_{s_0}, and so it specifies the Turing machines $S'_{i,D}, W, R'_{s_0}$.

Conditionally on $Sound(S')$ if:

$$*\, S'_{0,D}(\Sigma_0) \to Y = (X, \Sigma^1) \in \mathcal{T} \times Strings, \tag{3.2}$$

then

$$*X(\Sigma^1) \to A'_{s_0}(\Sigma_0)$$

if it halts. Then

$$*R'_{s_0}(Y, \Sigma_0) \to \infty$$

if it halts, otherwise we have a contradiction by (2.5). Thus S knows, conditionally on $Sound(S')$, by the above, that the statement $\mathcal{G}$:

$$*\, S'_{1,D}(\infty, \Sigma_0) \text{ is equivalent to } *\, A'_{s_0}(\Sigma_0), \tag{3.3}$$

holds. Now S asserts $Sound(S)$, hence if Θ_{Σ_0} then $Sound(S')$, and so S asserts $\mathcal{G}$ conditionally on Θ_{Σ_0}. This is the "weak Gödel statement" $\mathcal{G}$ we mentioned in the introduction, and structurally this part looks very much like the Penrose argument. However "weak Gödel statement" is just a name, we do not need any formal relationship with Gödel statements, although such a relationship could likely be extracted.

So far S has not run any computation on $\mathcal{C}$, he then runs $*S'_{1,D}(\infty, \Sigma_0)$.

Case 1. If $*S'_{1,D}(\infty, \Sigma_0)$ halts in time at most t_0, then S (more formally $\mathcal{A}$) prints:

$$S'_{1,D}(\infty, \Sigma_0) + 1,$$

that is

$$\mathcal{A}(\Sigma_0) = S'_{1,D}(\infty, \Sigma_0) + 1.$$

Then this printed value of $\mathcal{A}(\Sigma_0)$ contradicts $\Theta_{\Sigma_0} \wedge Sound(S')$, since if $\Theta_{\Sigma_0} \wedge Sound(S')$ then

$$\mathcal{A}(\Sigma_0) = S'_{1,D}(\infty, \Sigma_0) + 1 = \mathcal{A}'(\Sigma_0) + 1 = \mathcal{A}(\Sigma_0) + 1,$$

by (3.3), which is a contradiction.

Case 2. If $*S'_{1,D}(\infty, \Sigma_0)$ does not halt in time t_0, then S prints any integer "immediately" after this time t_0 has elapsed. S knows this will contradict Θ_{Σ_0} 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 $*\widetilde{S}^{s_0+1}_{1,D}(\infty, \Sigma_0)$ halts after at least

$$(s_0 + 1)t_0$$

time, when simulated on $\mathcal{C}$, since $*S'_{1,D}(\infty, \Sigma_0)$ halts in time at least t_0 on $\mathcal{C}$, and by construction of $\widetilde{S}^{s_0+1}_{1,D}$ in Lemma 2.10. Now if $\mathcal{C} = C_{s_0}$, then the halt time of $*A'_{s_0}(\Sigma_0)$ 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_0 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}, and Θ_{Σ_0} is disproved.

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 for this value of t_0 and s_0, 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.

Since Σ_0 was arbitrary our conclusion follows, at least assuming S was capable of making all the above deductions, but since we just did so, we may just assume this. □

The above theorem is a formal elaboration of our Theorem 0.1, 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 or objections

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?

Answer. 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. This leads us to the next question.

Question 3. Even if S asserts his fundamental soundness, given Σ he cannot conclude:

$$\mathit{Sound}(S) \text{ and hence } \mathit{Sound}(S') \text{ if } \Theta_\Sigma,$$

for S is aware of the fault issues of his biological brain, cf. the comments in the introduction on fundamental soundness. In particular $S'_{0,D}$ must also simulate these fault issues, which can lead to errors and exceptions. In this case the reasoning of S is faulty and hence it is not fundamental soundness that is contradicted by the argument, but only the faulty conclusion by S.

Answer. This "experiment" is controlled by E, she can iterate as much as necessary to control the issue of faults. Since S may also expect this, given that he believes himself to be fundamentally sound, $\mathit{Sound}(S')$ if Θ_Σ, in the context of a fixed Σ, can be expected with arbitrarily high confidence, after sufficiently many iterations. We can understand that this is what S means when he asserts $\mathit{Sound}(S)$. We can understand that this is what S means when he asserts $\mathit{Sound}(S)$. We can understand that this is what S means when he asserts $\mathit{Sound}(S)$. Adjusting for this would make our argument more involved, but only trivially so, and in this case the conclusion of S is certainly not obviously faulty. And to his defense, there is overwhelming evidence that mathematicians, when faced with a particular problem, stabilize after some time on sound solutions. So as in the previous question, even if we start accounting for faults and noise we can produce a contradiction to any requisite certainty.

5. Concluding remarks

The soundness hypothesis deserves much additional further study, far beyond what we can do here, and beyond what already appears in the work of Penrose, and others. Here is however one final amusing remark. If we are fundamentally unsound and computable, then given sufficient future advances in neuroscience and computer science, it will soon be possible to translate human brains to formal systems. Then computers should be able to, by brute force analysis, discover our inconsistencies, and they can then proceed to get us to assert in absolute faith that $0 = 1$. Moreover, in the context of our thought experiment above, we can say exactly where to look for the inconsistency, for if Θ_Σ, then the halting, computation $*S'_{0,D}(\Sigma)$ must involve such an inconsistency, and could then be analyzed.

Although it is possible that Turing computable artificial intelligence will start passing Turing tests, of the kind envisioned by Turin himself, given our theorem and the soundness hypothesis, 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.

Acknowledgements. Dennis Sullivan, David Chalmers and Bernardo Ameneyro Rodriguez, 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] 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.

[7] Adrian Kent. "Quanta and Qualia". In: *Foundations of Physics* 48.9 (Sept. 2018), pp. 1021–1037. ISSN: 1572-9516. URL: https://doi.org/10.1007/s10701-018-0193-9.

[8] Kobi Kremnizer and Andr/'e Ranchin. "Integrated Information-Induced Quantum Collapse". In: *Foundations of Physics* 45.8 (Aug. 2015), pp. 889–899.

[9] Chris Fields et al. "Conscious agent networks: Formal analysis and application to cognition". In: *Cognitive Systems Research* 47 (Oct. 2017).

[10] 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.

[11] 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.

[12] Roger Penrose. *Shadows of the mind*. 1994.

[13] Peter Koellner. "On the Question of Whether the Mind Can Be Mechanized, I: From Gödel to Penrose". In: *Journal of Philosophy* 115.7 (2018), pp. 337–360.

[14] Peter Koellner. "On the Question of Whether the Mind Can Be Mechanized, II: Penrose's New Argument". In: *Journal of Philosophy* 115.9 (2018), pp. 453–484.

[15] 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.

[16] David J. Chalmers. "Minds machines and mathematics". In: *Psyche, symposium* (1995).