# ANALOGUES OF GÖDEL STATEMENTS AND COMPUTABILITY OF INTELLIGENCE

YASHA SAVELYEV

Abstract. We show that there is a mathematical obstruction to complete Turing computability of intelligence. This obstruction can be circumvented only if human reasoning is fundamentally unsound. The most compelling original argument for existence of such an obstruction was proposed by Penrose, however Gödel, Turing and Lucas have also proposed such arguments. The ultimate argument of Penrose would undoubtedly work if a certain proposed Gödel statement could be constructed, however this statement may not even exist as the formal system in question is ill defined, so that we cannot use Gödel incompleteness theorem. We solve this problem by partially defining our subject just enough so that a certain analogue of a Gödel statement, or Gödel string as we call it in the language of Turing machines, can be constructed explicitly and directly, without use of Gödel incompleteness. 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 physical model.

*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 come in we need some assumption on the fundamental soundness of human reasoning. We need some qualifier like "fundamental" as even mathematicians are not on the surface sound at all times. In this note fundamental soundness is understood as follows. We are on the surface unsound not because of fundamental internal inconsistencies of our mental constructions, but for the following pair of reasons. First, due to time constraints humans make certain leaps of faith, without fully vetting their logic. Second, the noisy, faulty, biological nature of our brain leads to interpretation errors of our mental constructions. Here by "faulty", we mean the possibly common occurrence of faults in brain processes, coming from things like brain 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], based on existence of absolutely undecidable statements, see Feferman [4] for a discussion. Later Lucas [5] and later again and more robustly Penrose [6] argued for a no answer, based only on soundness. 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]

[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.

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 [7], 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: [8], [9], [10], [11].

The following is a slightly informal version of our main Theorem 4.1.

**Theorem 0.1.** *Either there are cognitively meaningful, non Turing computable processes in the human brain, or human beings are fundamentally unsound. Moreover, the actual soundness condition needed is very weak.*

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. The immediate implications and context of the above 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. [12]), 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 physical nature, 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 analysis.** What follows will be very close in essence to the argument Penrose gives in [13], which we take to be his main and final argument. However we partially reinterpret this 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 this language in our actual argument, which is based purely on the language of Turing machines, and is much more elementary. In fact, any misunderstandings of formal systems in the following outline of are likely due to me and not due to Penrose, I can only assert that the actual main argument formulated purely in Turing machines is sound.

Let $P$ be a human subject, which we understand at the moment as a machine printing statements in arithmetic, given some input. That is for each $\Sigma$ some string input in a fixed alphabet, $P(\Sigma)$ is a statement in arithmetic, e.g. "There are infinitely many primes." Say now $P$ is in contact with experimenter/operator $E$. The input strings $\Sigma_T$ that $E$ gives $P$ are pairs $(\Sigma_T, n)$ for $\Sigma_T$ specification of a Turing machines $T$, and $n \in \mathbb{N}$.

Let $\Theta_T$ be the statement:

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

For each $(T, n)$, $P$ prints his statement $P(\Sigma_T, n)$, which he asserts to hold if $\Theta_T$. We ask that for each $T$: $\{P(\Sigma_T, n)\}_n$ is the complete list of statements that he asserts to be true, conditionally on $\Theta_T$. If the reasoning powers of $P$ could be totally captured by some fixed formal system then all sentences in this formal system could be enumerated, so that the above supposition is allowable for the purpose of what follows.

Before we proceed, we put the condition on our $P$ that he asserts himself to be fundamentally sound. Let then $T_0$ be a specified Turing machine, and suppose that $E$ passes to $P$ input of the form $(\Sigma_{T_0}, n_0)$. Now $P$ reasons that since he is fundamentally sound, so that his deductions based on each $\Theta_T$ are sound, the same must be true of $T_0$ if $\Theta_{T_0}$. In particular he asserts:

$$\Theta_{T_0} \implies T_0(\Sigma_{T_0}, n), \tag{0.3}$$

for all $n$. On the other hand, if the statements $\Theta_T$ could be suitably formalized then the statements $\{T_0(\Sigma_{T_0}, n)\}_n$ must be the complete list of provable statements in a certain formal system $\mathcal{F}(T_0)$ associated to $T_0$, which would be sound if $\Theta_{T_0}$ and if (0.3), and in particular consistent. Then there would be a Gödel statement $G(T_0)$, which would be true if $\Theta_{T_0}$ and if (0.3) and implies that $T_0(\Sigma_{T_0}, n) \neq G(T_0)$, or all $n$. $P$ would then print $G(T_0)$. Now if $\Theta_{T_0}$ is true, and $P$ is indeed sound and hence (0.3) holds, then by construction $T_0$ cannot print $G(T_0)$, and so $T_0$ does not compute $P$, so we obtain a contradiction.

The above outline would at least in essence work if $G(T_0)$ was constructible by $P$. Unfortunately we cannot even be certain that such a statement exists, since $\Theta_T$ are not well defined statements since $P$ is not defined. Consequently $\mathcal{F}(T_0)$ at least a priori is not a well defined formal system, and we cannot use Gödel incompleteness theorem. We will delve no further into this. A full critique of the Penrose argument, is given in Koellner [14], [15], see also Penrose [16], and Chalmers [17] for discussions on related issues.

It might be possible for an expert logician to work out this problem directly, but we take a slightly different and far more elementary approach. To solve the above problem we partially define our subject henceforth denoted by $S$, by means of formalizing properties of a certain function associated to $S$. We do this so that a certain analogue of the Gödel statement can be constructed explicitly and directly. This is not exactly "Gödel statement" because we will not even be dealing with formal systems, but purely with Turing machines, in fact it would be more appropriate to call it a "Gödel string". But this string has analogous properties.

However, to partially formalize $S$ we need to work with a slightly stronger form of Turing computability of $S$. Roughly speaking, for us computability means that not only are the answers of $S$ as a function 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 mathematically explicit.) Of course this stronger computability is what one usually means when one talks of computability of intelligence, for us it just necessary to make it explicit.

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 $\Sigma$, the collection of symbols on the tape $T_i$, so that to the left and right of $\Sigma$ 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 $\Sigma_{out}$, the collection of symbols on the tape $T_o$, so that to the left and right of $\Sigma$ 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 $\Sigma$ 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***. In this case we denote by $s_f$ the final configuration, so that the sequence is eventually constant with terms $s_f$. We define the* **length** *of a computation sequence to be the first occurrence of $n > 0$ s.t. $s_n = s_f$. 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 $\Sigma$, 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.*

We will just call a partially defined function $f : Strings \to Strings$ as a function, for simplicity. So a Turing machine $T$ itself determines a function, 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. We do not ask that for $\Sigma \in I$ $*T(\Sigma)$ halts. If $*T(\Sigma)$ does halt then we shall say that $\Sigma$ 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 $\Sigma$ encodes an element of $Strings^2$ its components are $\Sigma_1$ and $\Sigma_2$." 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 implicitly encoded as subsets of *Strings*.

**Definition 1.6.** *We say that a Turing machine $T$ computes a function $f : Strings \to Strings$ 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 $\Sigma \in Strings^n$, with $i$'th component of $\Sigma$ 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 $\sigma_c^1, \sigma_o^1$ 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 $\sigma_c^2, \sigma_o^2$ 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 $(\Sigma_1, \Sigma_2)$, with $\Sigma_1$ an input string for $M_1$, and $\Sigma_2$ 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 $\Sigma_o^1$ and $T_o^2$ component $\Sigma_o^2$. Then the output $M_1 \star M_2(\Sigma_1, \Sigma_2)$ is defined to be $\Sigma_o^1$ if the final state is of the form $(q_f, q, 1)$ for $q_f$ final, or $\Sigma_o^2$ 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 $(T, \Sigma)$ for $T$ an encoding of a Turing machine and $\Sigma$ input to this $T$. It can be partially characterized by the property that for every Turing machine $T$ and $\Sigma$ 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 a function

$$T : I \to O,$$

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

When we intend to suppress dependence of a variable $V$ on some parameter $p$ we often write $V = V(p)$, this equality is then an equality of notation not of mathematical objects.

## 2. Setup for the proof of Theorem 0.1

**Definition 2.1.** *A* **machine** *is a (partially defined) function $A : Strings \to Strings \times \mathbb{N}$. The second component of $A(\Sigma)$ will be called* **time to answer** *on input $\Sigma$. In practice we will often omit to write the second component explicitly.*

Given a Turing machine $T : Strings \to Strings$, and a computer $C$ we have an associated machine

$$T_C : Strings \to Strings \times \mathbb{N},$$

with the $\mathbb{N}$ component $T_C^{\mathbb{N}}(\Sigma)$: the time that it takes the computation sequence $*T(\Sigma)$ to halt when simulated on $C$.

*Remark* 2.2. 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 just mean a computer simulation, that is a program running on a computer.

**Definition 2.3.** *We say that a machine $A$, is* **strongly computable** *if there exists a Turing machine $A'$, and a computer $C$ such that the corresponding machines $A$, $A'_C$ coincide (including time's to answer), on any input $\Sigma$ such that $A(\Sigma)$ is defined. If $C$ is as above we say that $A$ is* **strongly computed** *by $A'$ on $C$.*

For example suppose that the time to answer of each $A(\Sigma)$ is bounded in $\Sigma$, and $\pi_1 \circ A$ is computed by a Turing machine $A'$, for $\pi_1 : Strings \times \mathbb{N} \to Strings$ projection to first component. Suppose however that for any such $A'$ the length of the halting computation sequence $*A'(\Sigma)$ is unbounded in $\Sigma$. In this case $A$ cannot be strongly computable.

2.1. **Diagonalization machines.** Let now $\mathcal{T}$ denote the set of Turing machines with sets of inputs and outputs encoded as subsets of *Strings*, see Section 1 for what this means exactly. And let $\mathcal{M}_0$ denote the set of machines $M$ with input $\mathcal{T}$ and output in $\mathcal{T} \times \mathbb{Z}$. As we are going to directly construct a certain Turing machine analogue of a Gödel statement, to make it exceptionally simple we ask that elements of $\mathcal{M}_0$ have the following form, which expresses the property that these machines are "diagonalizing" against the input.

Note that in what follows we suppress the "time to answer" $\mathbb{N}$ component. For $M \in \mathcal{M}_0$ we can write it as a composition of machines:

$$M(T) = S_{1,D}(R(S_{0,D}(T), T), T), \tag{2.4}$$

where

$$S_{0,D} : \mathcal{T} \to \mathcal{T} \times Strings,$$
$$R : \mathcal{T} \times Strings \times \mathcal{T} \to \mathbb{Z} \sqcup \{\infty\},$$

where $\{\infty\}$ is the one point set containing the symbol $\infty$, which is just a particular distinguished symbol, also implicitly encoded as an element of *Strings*. And where

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

satisfies

$$S_{1,D}(x, T) = (S_{0,D}(T), S^{\mathbb{Z}}_{1,D}(x, T)) \tag{2.5}$$

for

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

satisfying:

$$S^{\mathbb{Z}}_{1,D}(x, T) = x^{\mathbb{Z}} + 1 \text{ if } x \in \mathbb{Z} \times \mathcal{T} \subset (\mathbb{Z} \sqcup \{\infty\}) \times \mathcal{T}, \tag{2.6}$$

where $x^{\mathbb{Z}}$ is the $\mathbb{Z}$ component of $x$.

We set $\mathcal{T}_0 \subset \mathcal{M}_0$ to be the subset corresponding to Turing machines with component machines $S_{i,D}, R$ likewise Turing machines.

**Definition 2.7.** *We say that a machine $M \in \mathcal{M}_0$ is* **totally computed** *by a $M' \in \mathcal{T}_0$ if $M$ is strongly computed by $M'$ and moreover $S_{i,D}$ is strongly computed by Turing machines $S'_{i,D}$ and $R$ is strongly computed by $R'$, for $S_{i,D}, R$ and $S'_{i,D}, R'$ the components of $M$ respectively $M'$ as above. Meaning that*

$$M(T) = S_{1,D}(R(S_{0,D}(T), T), T),$$

*and*

$$M'(T) = S'_{1,D}(R'(S'_{0,D}(T), T, T).$$

Given $M \in \mathcal{M}_0$ and $M' \in \mathcal{T}_0$ let $\Theta_{M,M'}$ be the statement:

$$M \text{ is totally computed by } M'. \tag{2.8}$$

**Definition 2.9.** *We say that $M \in \mathcal{M}_0$ is* **$P$-sound** *if for each $T \in \mathcal{T}$, with the output string $M(T) = (X, \Sigma^1, n)$ defined, the output string has property $O$, saying that:*

$$\Theta_{M,T} \implies *X(\Sigma^1) \text{ is equivalent to } *T(T).$$

*Here $T(T)$ really means $T(\Sigma_T)$ for $\Sigma_T$ the string encoding of the Turing machine $T$.*

Define a $P$-sound $M' \in \mathcal{T}_0$ analogously.

**Definition 2.10.** *If $M, M'$ as above are $P$-sound we will say that sound$(M)$, sound$(M')$ hold.*

For example a trivially $P$-sound machine $M$ is one for which $S_{0,D}(T) = (T, \Sigma_T)$ for every $T$.

**Definition 2.11.** *We say that $M \in \mathcal{M}_0$ has* **property** *$P$ if it is $P$-sound and if whenever $M(M')$ is defined for $M' \in \mathcal{T}_0$ then:*

$$\Theta_{M,M'} \implies R'(S'_{0,D}(M'), M') \neq \infty.$$

**Theorem 2.12.** *If $M$ has property $P$, then for every $M' \in \mathcal{T}_0$ if $\Theta_{M,M'}$ and $M(M')$ is defined then*

$$M(M') \neq (S'_{1,D}, (\infty, M'), n)$$

*for any $n$, where $S'_{1,D}$ is the component machine of $M'$ as above. On the other hand, clearly if $\Theta_{M,M'}$ then $*S'_{1,D}(\infty, M')$ is equivalent to $*M'(M')$, so that if the latter halts then $S'_{1,D}(\infty, M')$ has property $O$.*

So given a certain $M \in \mathcal{M}_0$ printing strings with property $O$, and given any $T \in \mathcal{T}_0$ such that $*T(T)$ halts, if $M$ is totally Turing computable by $T$, there is a certain explicitly constructible string $\mathcal{G}(T)$ with property $O$ s.t. $\mathcal{G}(T) \neq M(T)$. This "Gödel string" $\mathcal{G}(T)$ is what we are going to use. The proof is another diagonalization argument.

*Proof.* Suppose not and let $M'_0$ be such that $\Theta_{M,M'_0}$ and $M(M'_0) = (S'_{1,D}, (\infty, M'), n)$ for some $n$. Then

$$M(M'_0) = S'_{1,D}(\infty, M'_0)$$

since $M$ is $P$-sound and we have, after denoting by $M^{\mathbb{Z}}$ the $\mathbb{Z}$ component of the function:

$$\begin{aligned}
(S'_{1,D})^{\mathbb{Z}}(\infty, M'_0) = n &= M^{\mathbb{Z}}(M'_0) \\
&= (S'_{1,D})^{\mathbb{Z}}(R'(S'_{1,D}, (\infty, M'), M'_0)) \quad \text{by (2.5)} \\
&= (S'_{1,D})^{\mathbb{Z}}(S'_{1,D}(\infty, M'_0)) \quad \text{by property } P \\
&= n + 1 \quad \text{by (2.6)}.
\end{aligned} \tag{2.13}$$

So we obtain a contradiction. □

## 3. A system with a human subject $S$ as a machine with property $P$

Let $S$ be in an isolated environment, in communication with an experimenter/operator $E$ that as input passes to $S$ specifications of Turing machines. $S$ has in his environment a general purpose (Turing) digital computer, with arbitrarily, as necessary, expendable memory. $A$ will denote the above system: $S$ and his computer $\mathcal{C}$ in isolation. Here ***isolated environment*** means primarily that no information i.e. stimulus, that is not explicitly controlled by $E$ and that is usable by $S$, passes to $S$ while he is in this environment.

We suppose that upon receiving receiving a specification of a Turing machine $T$, $S$ implements the following protocol:

- Given $T$, $S$ decides after a time
$$t^0_D = t^0_D(T)$$
to run some computation $* = *_T$ on $\mathcal{C}$.
- $S$ then waits for a time
$$t^W = t^W(T)$$
for $*$ to halt.
- $S$ receives the output of $*$, or he stops waiting before it halts.
- $S$ decides after a time
$$t^1_D = t^1_D(T),$$
on his printed answer to $E$, which is unambiguously interpreted as an element of $\mathcal{T} \times Strings \times \mathbb{Z}$.

The initial "decision map" of $S$ may be understood as a machine:

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

suppressing the "time to answer" $\mathbb{N}$ component of the machine as is usual for us. The output $S_{0,D}(T)$ is a pair $(X, \Sigma^1)$ of a Turing machine $X$ and input $\Sigma^1$ 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 $S_{0,D}(T)$ is a pair of a computer program and input for this program that $S$ will run on $\mathcal{C}$.

We have another machine:

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

$R_{\mathcal{C}}(X, \Sigma^1, T)$ 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 $T$ as well, because how long he chooses to wait may depend on $T$. We set

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

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

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

if he does.

Likewise

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

denotes the machine representing 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 $S_{0,D}$. We suppose that for any $T \in \mathcal{T}$ this map satisfies:

$$S_{1,D}(x, T) = x + 1 \text{ if } x \in \mathbb{Z}. \tag{3.1}$$

We understand the "wait operation" by $S$ as a machine:

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

or

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

suppresing the $\mathbb{N}$ component. Here $\{\infty\}$ is the one point set as above, so that the first component of the output is only symbolic.

**Lemma 3.2.** *If $W$ is strongly computable, then $R_{\mathcal{C}}$ is likewise strongly computable.*

*Proof.* Suppose that $W$ is strongly computed by $W'$ on $C_1$, so that for each $(X, \Sigma^1, T)$ as above $*W'(X, \Sigma^1, T)$ halts in time $t^W(T)$. 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 $C_1$, meaning that the time to halt of any fixed computation run on $C_s$ is $\frac{1}{s}$ the time to halt on $C_1$.

For

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

if $\mathcal{C} = C_s$, we set

$$R'_s(Y, T) = W' \star U^s((Y, T), Y),$$

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

$$R'_s(Y, T) = X(\Sigma^1),$$

otherwise if it halts with a final state of $W$ then

$$R'_s(Y, T) = \infty.$$

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

Thus the above system $A$ and the protocol determine a machine:

$$A(T) = S_{1,D}(R_{\mathcal{C}}(S_{0,D}(T), T), T), \tag{3.3}$$

so that $A \in \mathcal{M}_0$.

In the above we only described the general protocol for $A \in \mathcal{M}_0$. We now consider a more specific $A_0$ of the type above, corresponding to a certain subject $S_0$, which behaves in the following defined way. In the case of this machine $A_0$, given any $A' \in \mathcal{T}$ our $S_0$ first checks its specification and we then have two cases.

*Case 1.* $A'$ is verbatim specified to be in $\mathcal{T}_0$, in particular has the form:

$$A'(T) = S'_{1,D}(R'(S'_{0,D}(T), T), T).$$

In this case $S$ chooses his computation to be:

$$*S'_{1,D}(\infty, A')$$

that is $S_{0,D}(A') = (S'_{1,D}, (\infty, A'))$ and his wait time is some $t^W > 0$. If this computation does not halt in time $t^W$ he prints $(S'_{1,D}, (\infty, A'), n)$ with arbitrary $n$, taking any time less then $t^W$ to do so. In other words the answer time of $S_{1,D}(\infty, A')$ is less then $t^W$. If $*S'_{1,D}(\infty, A') \to x$ then he prints

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

*Case 2.* $A'$ is not verbatim specified to be in $\mathcal{T}_0$, this includes the case where $A'$ could be rewritten/refactored in the form of an element in $\mathcal{T}_0$, but is not verbatim specified to be in $\mathcal{T}_0$. So we avoid having $S_0$ decide when some $T \in \mathcal{T}$ could be refactored to be in $\mathcal{T}_0$, which is likely undecidable, and is not necessary for our argument. In this case $S$ takes his computation to be $*A'(A')$, but otherwise proceeds exactly as above.

3.1. **The conditional $\Theta_{A,A'}$.** We now adjust our conditional $\Theta_{A,A'}$ to the following:

$$\begin{aligned}
&A \text{ is totally computed by } A',\\
&S_{i,D} \text{ are strongly computed on } C_1,\\
&\mathcal{C} = C_s \text{ with } s \text{ at least one.}
\end{aligned}$$

This of course adjusts the meaning of *sound*$(A)$, and property $P$, as defined following the Definition 2.9, but otherwise the statement of Theorem 2.12 remains unchanged.

## 4. Proof of Theorem 0.1

**Theorem 4.1.** *Let $A_0$ be the machine described above, then for every $A' \in \mathcal{T}$ either not $\Theta_{A_0,A'}$ or not sound$(A_0)$.*

*Proof.* Given $A' \in \mathcal{T}_0$, we first verify:

$$\text{if } \Theta_{A_0,A'} \text{ then } R'(S'_{0,D}(A'), A') \neq \infty.$$

Suppose by contradiction that $\Theta_{A_0,A'}$, and $R'(S'_{0,D}(A'), A') = \infty$, then $S'_{1,D}(\infty, A')$ halts in time less then $t^W$ on $C_1$, since $S$ takes less then $t^W$ time to print his final answer by assumption, and since $S'_{1,D}$ strongly computes $S_{1,D}$ on $C_1$. But $\mathcal{C} = C_s$ with $s$ at least 1, and $S$ waits for time $t^W$ for $*S'_{1,D}(\infty, A')$ to halt on $\mathcal{C}$, so this would have halted during his wait period, so that $R'(S'_{0,D}(A'), A') \neq \infty$, and we have a contradiction. Suppose *sound*$(A_0)$, then by the above $A_0$ has property $P$.

Now suppose also $\Theta_{A_0,A'}$ for some $A'$, then we may assume that $A'$ is explicitly specified as an element of $\mathcal{T}_0$, since $A_0$ is explicitly specified to be in $\mathcal{M}_0$. Then by Theorem 2.12

$$A_0(A') \neq (S'_{1,D}, (\infty, A'), n)$$

for any $n$, since $A_0$ has property $P$. But this contradicts the defining property of $A_0$, so one of our assumptions has to be wrong. □

The above theorem is a formal elaboration of our Theorem 0.1, if our subject $S_0$ asserts that he is fundamentally sound, which means in this context that he asserts in absolute faith that $sound(A_0)$ holds.

□

## 5. A possible objection

*Objection* 1. Even if $S_0$ asserts his fundamental soundness, given $A' \in \mathcal{T}_0$ he cannot conclude with certainty that his printed string $A_0(A')$ has property $O$. For $S$ must be aware of the fault issues of his biological brain, cf. the comments in the introduction on fundamental soundness. So $S_0$ could only assert that $A_0(A')$ has property $O$ sometimes, depending on the "noise" conditions of his brain, that is depending on how likely he is to fault.

*Answer.* There are at least a couple of ways to answer this. We could say that this particular $S_0$ does not believe that "brain noise" can be an issue here, because the soundness condition required of him is very weak.

On the other hand suppose $S_0$ and we do take the noise objection seriously. Then to make better sense of this objection we need to decorate our machine $A_0$ with time $t \in \mathbb{N}$, so that $A_0^t(A')$ signifies that this is the $t$'th time that $A_0$ is asked to print its answer given $A'$. So the machine $A_0^t$ is in principle $t$ dependent. But clearly we don't actually need that $A_0^t(A')$ has property $O$ for every $t$, only that there is a non-zero, bounded from below, uniformly for all $t$, probability of this happening. We may understand that this is what $S_0$ means when he asserts $sound(A_0)$. Given our interpretation of fundamental soundness we may clearly suppose this, at least up to interpretation - we cannot take $t$ to infinity given the finite life span of $S_0$. Essentially the same argument as in the proof of Theorem 4.1 would then give our conclusion with probability 1.

## 6. 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 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 discover our inconsistencies by brute force analysis. 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 $\Theta_{A_0,A'}$ then the halting computation $*S'_{0,D}(A')$ must involve such an inconsistency, and could then be analyzed.

There is still an door open for non Turing computable simulations of human 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, Bernardo Ameneyro Rodriguez, and Dusa McDuff 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] S. Feferman. "Are There Absolutely Unsolvable Problems? Godel's Dichotomy". In: *Philosophia Mathematica* 14.2 (2006), pp. 134–152.
[5] J.R. Lucas. "Minds machines and Goedel". In: *Philosophy* 36 (1961).
[6] Roger Penrose. *Emperor's new mind.* 1989.

[7] 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.

[8] 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.

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

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

[11] 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.

[12] 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.

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

[14] 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.

[15] 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.

[16] 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.

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