TURING 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. We first slightly reformulate the argument of Penrose keeping the main idea the same. In this formulation we argue that his argument works up to possibility of construction of a certain Gödel statement. We then completely re-frame the argument in the language of Turing machines, and by partially defining our subject just enough, we show that a certain analogue of a Gödel statement, or a Gödel string as we call it in the language of Turing machines, can be readily constructed directly. 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. Let us call all these possible fault vectors "brain nose". In other words, according to us to say that a human being is fundamentally sound, is to say that after "stripping out" the "brain noise" this human being will be sound, and have the same reasoning powers.

Gödel first argued for a no answer in [3, p. 310], relating the question to existence of absolutely undecidable problems, see Feferman [4] for a discussion. Since existence of absolutely undecidable problems is such a difficult and contentious issue, even if Gödel's argument is in essence correct it is not completely compelling.

Later Lucas [5] and later again and more robustly Penrose [6] argued for a no answer, based only on soundness. Such an argument if correct would be extremely compelling. 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 [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. This theorem is indeed a mathematical fact, given our chosen interpretation of fundamental soundness of human reasoning as outlined above.*

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 particular any gaps of the following outline should disappear.

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 Σ some string input in a fixed finite 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 that E gives P are pairs (Σ_T, n) for Σ_T specification of a Turing machines T, and $n \in \mathbb{N}$.

Let Θ_T be the statement:

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

For each (Σ_T, n), P prints his statement $P(\Sigma_T, n)$, which he asserts to hold if Θ_T. We ask that for each T: $\{P(\Sigma_T, n)\}_n$ is the complete list of statements that P asserts to be true conditionally on Θ_T. Finally, we put the condition on our P that he asserts himself to be fundamentally consistent. More specifically P asserts the statement I_{T_0}:

$$\Theta_T \implies T \text{ is consistent.} \tag{0.3}$$

Asserting ones own consistency is by no means contentious as most people assert their consistency, in some form, by implication as follows. For if a human H asserts $0 \neq 1$ in absolute faith, that is H asserts that they will never assert $0 = 1$, while "sane", then by implication H asserts their consistency. For if H is not consistent they must eventually assert everything, while "sane", in particular $0 = 1$.

Let then T_0 be a specified Turing machine, and suppose that E passes to P input of the form (Σ_{T_0}, n_0). Now, as is well known, 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 consistent if Θ_{T_0} and if I_{T_0}. In particular if Θ_{T_0} and if I_{T_0}, then there would be a true (in the standard model of arithmetic)

[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.

Gödel statement $G(T_0)$ such that $T_0(\Sigma_{T_0}, n) \neq G(T_0)$, for all n. This is at least assuming some additional properties of P so that $\mathcal{F}(T_0)$ would satisfy the other condition of Gödel's theorem, (we need that $\mathcal{F}(T_0)$ could interpret certain arithmetic).

But P asserts I_{T_0}, hence he must assert by implication that

$$\Theta_{T_0} \implies G(T_0),$$

and so if P knew how to construct $G(T_0)$ then this statement must be in the list $\{P(\Sigma_{T_0}, n)\}_n$, and so in the list $\{T_0(\Sigma_{T_0}, n)\}_n$, so we would get a contradiction. So either not Θ_{T_0} or P is not consistent.

The above outline would at least in principle work if $G(T_0)$ was constructible by P. From this author's point of view, at least if the hypothesis of Gödel's theorem are satisfied by $\mathcal{F}(T_0)$, then constructibility of $G(T_0)$ is not in principle an obstruction. This is because the specification of $\mathcal{F}(T_0)$ could at least in principle be explicitly obtained by P, given the finite specification of T_0. And the Gödel statement could always, at least in principle, be explicitly constructed once one knows the formal system, even if in practice this may be hopelessly difficult. We will delve no further into this. One detailed critique of the Penrose argument is given in Koellner [14], [15], see also Penrose [16], and Chalmers [17] for discussions on related issues. (Note of course that our argument above is a bit different.)

In this note we completely solve the above problem of explicit construction of the Gödel statement. We reformulate the above idea using a far more elementary approach, more heavily based in Turing machines. Next, 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 readily constructed directly, avoiding intricacies of formal systems and the general incompleteness theorem. This will not be exactly "Gödel statement", but rather a "Gödel string" as we call it, because we will not even be dealing with formal systems, but purely with Turing machines. But this string has analogous properties.

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, (input, output and computation) 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 they point. 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 $F \subset Q$ 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_{out} 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*

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

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**. *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 Σ, if the associated Turing computation $*M(\Sigma)$ halts.

Definition 1.3. *Let Strings denote the set of all finite strings, including the empty string $\emptyset$, 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 may 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 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 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 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 implicitly encoded as subsets of $Strings$.

Definition 1.6. *We say that a Turing machine T computes a function $f : I \to J$, if I is contained in the set of permissible inputs of T and $*M(\Sigma) \to f(\Sigma)$, whenever $f(\Sigma)$ is defined, for $\Sigma \in I$.*

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

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

1.3. **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** *will be a synonym for a partially defined function $A : I \to O$, with I, O abstract sets with a prescribed encoding as subsets of Strings. (cf. Preliminaries.)*

$\mathcal{M}$ will denote the set of machines. Given a Turing machine $T : I \to O$, we have an associated machine T by forgetting all structure except the structure of a partially defined function. $\mathcal{T}$ will denote the set of machines, which in addition have a structure of a Turing machine.

2.1. **Diagonalization machines.** We denote by $\mathcal{T}_{\mathbb{Z}} \subset \mathcal{T}$ the subset of Turing machines, with output in $\mathbb{Z}$. Let $\mathcal{M}_0$ denote the set of machines M with input $\mathcal{I} = \mathcal{T} \times \mathbb{N}$ and output in $\mathcal{T}_{\mathbb{Z}} \times Strings$. As we are going to directly construct a certain Turing machine analogue of a Gödel statement, to make it exceptionally simple we will need to formulate some specific properties for our machines that will require a bit of setup.

For $M \in \mathcal{M}$, $(B, m) \in \mathcal{I}$ we set

$$D_0(B, m) = (M(B, m), m), \tag{2.2}$$

and we suppose that the range of D_0 is

$$I_0 \subset \mathcal{T}_{\mathbb{Z}} \times (Strings_0 = Strings \times \mathbb{N}),$$

consisting of (X, Σ) for Σ a permissible input for X.

Let

$$D_1 : \mathbb{Z} \sqcup \{\infty\} \to \mathbb{Z},$$

be a Turing machine which satisfies

$$D_1(x) = x + 1 \text{ if } x \in \mathbb{Z} \subset \mathbb{Z} \sqcup \{\infty\} \tag{2.3}$$

$$D_1(\infty) = 1. \tag{2.4}$$

And let

$$R : I_0 \subset \mathcal{T}_{\mathbb{Z}} \times Strings_0 \to \mathbb{Z} \sqcup \{\infty\},$$

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

We suppose that R satisfies the following property, which we call ***property*** G:

- R halts on the entire I_0.
- $R(X,\Sigma) \neq \infty \implies R(X,\Sigma) = X(\Sigma)$
- $R(D_1,\infty) \neq \infty$, and so $R(D_1,\infty) = 1$.

Lemma 2.5. *There is a Turing machine*

$$R : I_0 \subset \mathcal{T}_{\mathbb{Z}} \times Strings_0 \to \mathbb{Z} \sqcup \{\infty\},$$

satisfying property G.

Proof. Let W be some Turing machine $W : \{\emptyset\} \to \{\infty\}$, for $\emptyset$ the empty string. So as a function it is not very interesting since the input and output are singletons. We ask that the length of $*W(\emptyset)$ is $n > 0$, (cf. Preliminaries).

For

$$Y = (X,\Sigma) \in I_0$$

set

$$R_n(X,\Sigma) = W \star U(\emptyset,(X,\Sigma)),$$

in the language of the join operation described in Section 1, for U the universal Turing machine. Clearly R_n halts on the entire I_0, and satisfies

$$R_n(X,\Sigma) \neq \infty \implies R_n(X,\Sigma) = X(\Sigma).$$

As a function D_1 is completely determined but it could have various implementations as a Turing machine, so that the length l of $*D_1(\infty)$ depends on this implementation. We fix $n > l$, then by construction $R_n(D_1,\infty) = D_1(\infty)$. Thus $R := R_n$ has property G. □

We set $\mathcal{T}_0 \subset \mathcal{M}_0$ to be the subset corresponding to Turing machines, and $\mathcal{I}_0 = \mathcal{T}_0 \times \mathbb{N}$. Given $M \in \mathcal{M}_0$ and $M' \in \mathcal{T}_0$ let $\Theta_{M,M'}$ be the statement:

$$M \text{ is computed by } M'. \tag{2.6}$$

Definition 2.7. *For $M \in \mathcal{M}_0$, $M' \in \mathcal{T}_0$, an abstract element $(X,\Sigma_1) \in \mathcal{T}_{\mathbb{Z}} \times Strings$ is said to have* **property** $C = C(M,M')$ *if*

$$\Theta_{M,M'} \implies \forall m \quad (X(\Sigma_1,m) = D_1 \circ R \circ D_0'(M',m)) \vee (*M'(M',m) \text{ does not halt}),$$

for D_0' determined by M' as in (2.2).

To be more formal when we write $*M'(M',m)$ we should write $*M'(\Sigma_{M'},m)$ for $\Sigma_{M'}$ the string encoding of the Turing machine M'. But we conflate the notation for the Turing machine and its string specification.

Definition 2.8. *We say that $M \in \mathcal{M}_0$ is* **C-sound** *if for each $T = (M',m) \in \mathcal{I}$, with $M(T) = (X,\Sigma_1)$ defined, (X,Σ_1) has property $C(M,M')$. Thus $M \in \mathcal{M}_0$ is C-sound iff it prints strings with property C, which expresses a certain "diagonalization" property with respect to the input, in the sense that something analogous to Cantor's diagonalization is happening.*

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

Definition 2.9. *If M, M' as above are C-sound we will say that sound(M), sound(M') hold.*

Example 1. A trivially C-sound machine M is one for which $M(M',m) = (D_1 \circ R \circ D_0', M')$ for every $(M',m) \in \mathcal{I}$. In general, for any M, M' the list of all strings $\{(X,\Sigma_1)\}$ with property $C(M,M')$ is always infinite, as by example above there is at least one such string, which can then be modified to produce infinitely many such strings.

In what follows we understand D_1 as an element of $\mathcal{T}_{\mathbb{Z}}$, denoting the Turing machine:

$$(x,m) \mapsto D_1(x),$$

for $(x,m) \in (\mathbb{Z} \sqcup \{\infty\}) \times \mathbb{N}$.

Theorem 2.10. *If sound*$(M) \wedge \Theta_{M,M'}$ *then*

$$\forall m \quad M(M', m) \neq (D_1, \infty).$$

On the other hand, if sound(M) *then for any* $M' \in \mathcal{T}_0$ *the string*

$$\mathcal{G} = (D_1, \infty)$$

has property $C(M, M')$.

So given any C-sound $M \in \mathcal{M}_0$ there is a certain string $\mathcal{G}$ with property $C(M, M')$ for all M', s.t. for each M' if $\Theta_{M,M'}$ then

$$\mathcal{G} \neq M(M', m)$$

for all m. This "Gödel string" $\mathcal{G}$ is what we are going to use further on. What makes $\mathcal{G}$ particularly suitable for our application, is that it is independent of the particulars of M, M', all that is needed is $\mathcal{M} \in \mathcal{M}_0$ and is C-sound.

Proof. Suppose not and let M_0' be such that $\Theta_{M,M_0'}$ and

$$M(M_0', m) = (D_1, \infty) \text{ for some } m \ . \tag{2.11}$$

Then if $T = (M_0', m)$ we have

$$\begin{aligned} 1 = D_1(\infty) = D_1 \circ R \circ D_0'(T) &= D_1 \circ R(D_1, \infty) \\ &= 2 \text{ by property } G \text{ of } R \text{ and (2.3)}. \end{aligned}$$

The first line is since M is C-sound, since $\Theta_{M,M_0'}$, and since $*M_0'(T)$ halts since $M(T)$ is defined. So we obtain a contradiction.

We now verify the second part of the theorem. Given $T = (M', m) \in \mathcal{T}_0$, we show that:

$$\forall m \quad sound(M) \wedge (*M'(T) \text{ halts}) \wedge \Theta_{M,M'} \implies R(D_0'(T)) = \infty. \tag{2.12}$$

Suppose otherwise that

$$R(D_0'(T)) \neq \infty, \tag{2.13}$$

then since $M'(T) = (X, \Sigma_1)$ is defined and since R is everywhere defined

$$R(D_0'(T)) = x = X(\Sigma_1, m) \in \mathbb{Z} \text{ by Property } G \text{ and by (2.13)}.$$

Then we get:

$$x = X(\Sigma_1, m) = D_1 \circ R \circ D_0'(T) = D_1(x) = x + 1$$

by $sound(M)$, $\Theta_{M,M'}$ and by (2.3), so we get a contradiction and (2.12) follows. Our conclusion readily follows.

□

3. A system with a human subject S as a machine in $\mathcal{M}_0$

Let S be in an isolated environment, in communication with an experimenter/operator E that as input passes to S elements of $\mathcal{I} = \mathcal{T} \times \mathbb{N}$. S has in his environment a general purpose digital computer with arbitrarily, as necessary, expendable memory, (in other words a universal Turing machine). A will denote the above system: S and his computer $\mathcal{C}$ in the isolated environment. 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 T, as a specified program in his computer, after possibly using his computer $\mathcal{C}$ in some way, S instructs his computer to print after some indeterminate time an element

$$A(T) \in \mathcal{T}_{\mathbb{Z}} \times Strings,$$

so that we have a machine $A \in \mathcal{M}_0$.

Definition 3.1. *We say that S as above is* **computable** *if A is computable.*

3.1. **Additional conditions.** We now consider a more specific A_0 of the type above, corresponding to a certain subject S_0, which additionally behaves in the following way. For any fixed $B \in \mathcal{T}_0$

$$\{A_0(B, m)\}_m$$

is the complete list of strings that S_0 asserts to have property $C(A_0, B)$. Of course we don't actually need to S_0 to list infinitely many strings, we only need that S_0 can list as many strings as we like, and that given any particular B, eventually any particular string that S_0 asserts to have property $C(A_0, B)$ will appear. Also as in the Penrose argument we ask that S_0 asserts that he is fundamentally sound, meaning in this case that he asserts $sound(A_0)$.

4. Proof of Theorem 0.1

Theorem 4.1. *Let A_0 be the machine described above, then*

$$A_0 \text{ is computable } \implies \neg sound(A_0).$$

That is if our subject S_0 is computable he cannot be fundamentally sound.

This formalizes Theorem 0.1.

Proof. Suppose that for some $A' \in \mathcal{T}_0$ $\Theta_{A_0,A'} \wedge sound(A_0)$, then by Theorem 2.10

$$A_0(A', m) \neq (D_1, \infty)$$

for any m. On the other hand S_0 asserts $sound(A_0)$ and hence must assert that (D_1, ∞) has property C, by the second half of Theorem 2.10. In particular the string (D_1, ∞) must be in the list $\{A_0(A', m)\}_m$, since this list is meant to be complete. So we have reached a contradiction. □

5. A possible objection

Objection 1. Even if S_0 asserts his fundamental soundness, he cannot conclude $sound(A_0)$. For S_0 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 $A_0(A', m)$ has property C sometimes, depending on the "noise" conditions of his brain, that is depending on how likely he is to fault.

Answer. Our human subjects S are meant to be idealized. So that all "brain noise" issues are stripped out of them. By our definition of fundamental soundness in the introduction this cannot affect the reasoning power of such an S. This idealization may in principle be made more concrete by assuming that our S is wired up to a machine which detects faults due to the above mentioned "noise". This machine may just be a computer proof system of some kind. Indeed our system A above already has a computer, which is partly meant to reduce faults of S.

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 computable, then given sufficient future advances in neuroscience and computer science, it will soon be possible to translate human brains to formal systems, which are necessarily inconsistent by our theorem. (We are conflating soundness and consistency here, but this can be justified in the present context.) Then computers should be able to discover our inconsistencies by brute force analysis of these formal systems. 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 in principle empirically discover such an inconsistency as follows. $A_0(A', m_0)$ for some m_0 must *provably* not have property $C(A_0, A')$, since if S_0 is inconsistent he must eventually assert that every element of $\mathcal{T}_{\mathbb{Z}} \times Strings$ has property $C(A_0, A')$. Then if $\Theta_{A_0,A'}$, the computation $*A'(A', m_0)$ can be analyzed to discover an inconsistency.

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] 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).