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, with the latter formally interpreted here as certain stable soundness. 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 review the main issues with this argument, as well as outline a partial direct fix. We then completely re-frame the argument in the language of Turing machines, and by 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, without appeal to the Gödel incompleteness theorem. One crucial upshot of this new formulation is that the above Gödel string works in the context of stable soundness, and not just soundness, and thus we eliminate the final objections.

In what follows we understand ***human intelligence*** very much like Turing in [1], purely as a machine, a black box which receives inputs and produces outputs. More specifically, this black box B is meant to be some system which contains a human subject. We do not care about what is happening inside B. So we are not directly concerned here with such intangible things as understanding, intuition, consciousness - all the things that are valued of humans, and are supposed as special. The only thing that concerns us is what output B produces given an input, it matters not in the present discussion how it is produced. Given this *very* limited interpretation, the question that we are interested in is this:

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

An informal definition of a Turing machine (see [2]) is as follows: it is an abstract machine which permits certain inputs, and produces outputs. The outputs are determined from the inputs by a fixed finite algorithm, in a specific sense. For a non-expert reader we point out that this does not preclude the algorithm from "learning", it just means that how it "learns" is completely determined by the initial fixed algorithm. In particular anything that can be computed by computers as we know them can be computed by a Turing machine. For our purposes the reader may simply understand a Turing machine as a digital computer with unbounded memory running any particular 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", [1], 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 or consistency of human reasoning. Informally, soundness here means here that when our human asserts something in absolute faith, this something is indeed true. This requires context, as truth in general is undefinable. For our arguments later on the context will be in certain mathematical models.

We need some qualifier like "fundamental" as even mathematicians are not on the surface sound at all times, they may assert mathematical untruths at various times, (but usually not in absolute faith). Our preliminary understanding of this "fundamental" is 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 physically noisy, faulty, biological nature of our brain may lead to interpretation errors of our mental constructions. By "faulty" we mean the possibly common occurrence of faults in brain processes, coming from things like brain cell death, signaling noise between neurons, neurotransmitter imbalance, etc. Let us call all these possible fault vectors "brain noise". 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 undiminished reasoning powers. Later on we formally interpret fundamental soundness in terms of a certain mathematically precise notion of stable soundness, although it will be limited to a very specific context.

Around the same time as Turing, Gödel argued for a no answer to Question 1, see [3, p. 310], relating the question to existence of absolutely undecidable problems, see also 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 also a partial list of some partially related work on mathematical models of brain activity and or quantum collapse models: [8], [9], [10], [11].

The arguments of Penrose and Lucas have well known issues, which we intend to completely resolve here. The following is a slightly informal version of our main Theorem 5.3.

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, after formally interpreting fundamental soundness as stable soundness, which is to be defined.*

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 fluid flow 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 they 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 independently, non-computable processes in physical nature, in fact in the functioning of the brain, or human beings are fundamentally unsound. 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.

The original argument of Penrose. Penrose has given variations of the argument for a no answer to Question 1 in his books [6], [13]. The final argument can be found in [14], and it goes roughly as follows. Loosely, a formal system consists of a language: alphabet and grammar, a collection of sentences in this language understood as axioms, and finally a deductive system. Given a formal system F the statement Θ_F:

$$\text{I am } F,$$

means that any statement in arithmetic that I assert to be true is provable in F, e.g. "There are infinitely many primes." may be such a statement. The statement Θ_F for an F satisfying certain properties is equivalent to me being computable as a machine printing statements in arithmetic. We will call such an F ***good***. [2] So we suppose from now on that F is good, since we are interested in computability first and for most.

[1] It is likely most mathematicians would sympathize with Gödel, after all the entirety mathematics is meaningless if mathematicians are fundamentally unsound.

[2] Explicitly, it is a condition for the axioms of $\mathcal{F}$ being recursively enumerable, plus another minor condition on being able to prove enough basic things about numbers.

Now I assert I am consistent, which entails more specifically that I assert:

$$\text{If } \Theta_F \text{ then } F \text{ is consistent.} \tag{0.2}$$

By F being consistent we just mean that the formal system F does not prove a statement and its logical negation. (0.2) is not yet a statement of arithmetic, but we will get there.

Remark 0.3. Asserting ones own consistency is not completely irrational, as most people assert their consistency in some form by implication. 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 (but accepts basic logic) they must eventually assert everything, while "sane", in particular $0 = 1$. Of course if we analyze this more deeply then it is not consistency that people assert but rather certain stable consistency, we will get back to this soon, and for now delve no further.

To continue with the argument, as I assert (0.2) then I also assert by implication I_F:

$$\text{If } \Theta_F \text{ then } G(F) \text{ holds.}$$

If I assert $G(F)$, then this would be a contradiction to either my consistency or to Θ_F, since $G(F)$ is something that F cannot prove. Unfortunately I cannot assert $G(F)$, since I don't know Θ_F. I only assert I_F, so there is no contradiction here. But we may fix this idea as follows. Unfortunately, for this fix we need to get a bit more technical.

Outline of a partial fix of the Penrose argument. While this outline uses some of the language of formal systems, we will *not* use this language in our main 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: an ordered collection of symbols chosen from 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:

$$T \text{ computes } P. \tag{0.4}$$

For each (Σ_T, n), P prints his statement $P(\Sigma_T, n)$, which he asserts to hold if Θ_T holds. We ask that for each fixed 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 consistent. More specifically, P asserts for each T the statement I_T:

$$\Theta_T \implies T \text{ is consistent.} \tag{0.5}$$

By T being consistent we mean here:

$$T(\Sigma_T, n) \neq \neg(T(\Sigma_T, m)),$$

for any n, m with $\neg$ the logical negation of the statement, and where inequality is just string inequality of the corresponding sentences.

Let then T_0 be a specified Turing machine, and suppose that E passes to P input of the form (Σ_{T_0}, n). Now, as is well known [3], the statements $\{T_0(\Sigma_{T_0}, n)\}_n$ must be the complete list of provable statements in a certain formal system $\mathcal{F}(T_0)$ explicitly constructible given T_0. And $\mathcal{F}(T_0)$ 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) Gödel statement $G(T_0)$ such that $T_0(\Sigma_{T_0}, n) \neq G(T_0)$, for all n.

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

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

[3] I don't know a standard reference but see for example [4].

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}, that is P is not computed by T_0 or P is not consistent, but T_0 is arbitrary so we obtain an obstruction to computability of P.

The above outline would at least in principle work if $G(T_0)$ was constructible by P. From this author's point of view constructibility of $G(T_0)$ is not in principle an issue. This is because the specification of $\mathcal{F}(T_0)$ could 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. The other objection to the above argument is that strictly speaking it only proves: either we are non-computable or inconsistent. But humans are clearly inconsistent, so what does this mean? Of course as we have argued we must talk of fundamental soundness/consistency, but then the argument cannot work exactly as above. We will delve no further into critiquing the Penrose argument. One critique of the Penrose argument is given in Koellner [15], [16], see also Penrose [14], and Chalmers [17] for discussions on related issues. (Note of course that our argument (the partial fix) above is significantly different, and so the issues are different.)

So motivated by the discussion above, the ideal thing to do, is to formally define fundamental soundness and construct a new type of Gödel statements, which works under this weaker hypothesis. This is actually what we will do, in the limited setting above. We completely solve both problems: formally defining fundamental soundness in terms of a certain notion of stable soundness, and explicit construction of the "Gödel statement". We reformulate the above idea using a more elementary approach, more heavily based in Turing machines. To this end, 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 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, and crucially it will work in the context of fundamental, or stable soundness.

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, 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 total states of M, determined by the input Σ and M, with s_0 the initial configuration whose internal state is q_0, s_{i+1} is the total state that results from executing the instructions set of M on the total state s_i. 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 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. We write $T_1(\Sigma_1) = T_2(\Sigma_2)$ if $*T_1(\Sigma_1)$, $*T_2(\Sigma_2)$ both halt with the same value.*

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, or just ***input set*** and O is the set output strings or ***output set***, 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 explicit 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 $*T(\Sigma) \to f(\Sigma)$, whenever $f(\Sigma)$ is defined, for $\Sigma \in I$.*

Given Turing machines
$$M_1 : I \to O, M_2 : J \to P,$$
we may naturally ***compose*** them to get a Turing machine $M_2 \circ M_1 : C \to P$, for $C = M_1^{-1}(O \cap J)$, ($O \cap J$ is understood as intersection of subsets of *Strings*). C can be empty in which case this is a machine which rejects all input. Let us not elaborate further 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 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. **Universality.** It will be convenient to refer to the universal Turing machine U. This is a Turing machine already appearing in Turing's [2], 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 the structure of a Turing machine.

2.1. **Diagonalization machines.** 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.

We denote by $\mathcal{T}_{\mathbb{Z}} \subset \mathcal{T}$ the subset of Turing machines of the type:

$$X : (S_X \times \mathbb{N} \subset Strings \times \mathbb{N}) \to \mathbb{Z}.$$

In other words, the input set of $X \in \mathcal{T}_{\mathbb{Z}}$ is of the form $S_X \times \mathbb{N}$, for $S_X \subset Strings$, and the output set of X is $\mathbb{Z}$.

Let $\mathcal{O} \subset \mathcal{T}_{\mathbb{Z}} \times Strings$ consist of $(X, \Sigma) \in \mathcal{T}_{\mathbb{Z}} \times Strings$ with $\Sigma \in S_X$, defined as above. And set

$$\mathcal{O}' := \mathcal{O} \times \mathbb{N} \subset \mathcal{T}_{\mathbb{Z}} \times Strings \times \mathbb{N}.$$

Let

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

be a fixed Turing machine which satisfies

$$D_1(x) = x + 1 \text{ if } x \in \mathbb{Z} \subset \mathbb{Z} \sqcup \{\infty\} \tag{2.2}$$

$$D_1(\infty) = 1. \tag{2.3}$$

Here $\{\infty\}$ is the one point set containing the symbol ∞, which is just a particular distinguished symbol, also implicitly encoded as an element of *Strings*. In what follows we sometimes understand D_1 as an element of $\mathcal{T}_{\mathbb{Z}}$, denoting the Turing machine:

$$(x, m) \mapsto D_1(x), \tag{2.4}$$

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

We need one more Turing machine.

Definition 2.5. *We say that a Turing machine*

$$R : \mathcal{O}' \subset \mathcal{T}_{\mathbb{Z}} \times Strings \times \mathbb{N} \to \mathbb{Z} \sqcup \{\infty\},$$

has **property** G *if the following is satisfied:*

- *R halts on the entire $\mathcal{O}'$, that is $\mathcal{O}'$ is the set of R-acceptable strings.*
- *$R(X, \Sigma, m) \neq \infty \implies R(X, \Sigma, m) = X(\Sigma)$, for $(\Sigma, m) \in Strings \times \mathbb{N}$ and $X \in \mathcal{T}_{\mathbb{Z}}$.*
- *$\forall m : R(D_1, \infty, m) \neq \infty$, and so $\forall m : R(D_1, \infty, m) = 1$, by previous property.*

Lemma 2.6. *There is a Turing machine R satisfying property G.*

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

For

$$(X, \Sigma, m) \in \mathcal{O}'$$

set

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

in the language of the join operation described in Section 1, for U the universal Turing machine. Clearly R_n halts on the entire $\mathcal{O}'$, and satisfies

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

As a function $\mathbb{Z} \sqcup \{\infty\} \to \mathbb{Z}$, D_1 is completely determined but it could have various implementations as a Turing machine, so that the length l_m of $*D_1(\infty, m)$ depends on this implementation. Clearly we may assume that $\forall m : l = l_m$ for some l, by definition of D_1 as an element of $\mathcal{T}_{\mathbb{Z}}$, as in (2.4). We then ask that $n > l$ is fixed. Then by construction we get:

$$\forall m : R_n(D_1, \infty, m) = D_1(\infty, m) = 1.$$

Thus $R := R_n$ has property G. $\square$

Define $\mathcal{M}_0$ to be the set of machines M whose input set is $\mathcal{I} = \mathcal{T} \times \mathbb{N}$ and whose output set is *Strings*. We set $\mathcal{T}_0 \subset \mathcal{M}_0$ to be the subset corresponding to Turing machines, and we set $\mathcal{I}_0 := \mathcal{T}_0 \times \mathbb{N}$. Given $M \in \mathcal{M}_0$ and $M' \in \mathcal{T}_0$ let $\Theta_{M,M'}$ be the statement:

$$M \text{ is computed by } M'. \tag{2.7}$$

For each $M \in \mathcal{M}_0$, we define a machine:

$$\widetilde{M} : \mathcal{I} \to Strings \times \mathbb{N}$$

$$\widetilde{M}(B, m) = (M(B, m), m), \tag{2.8}$$

which is naturally a Turing machine when M is a Turing machine.

In what follows when we write $M'(M', m)$ we mean $M'(\Sigma_{M'}, m)$ for $\Sigma_{M'}$ the string encoding of the Turing machine M'. So we conflate the notation for the Turing machine and its string specification.

Definition 2.9. *For $M \in \mathcal{M}_0$, $M' \in \mathcal{T}_0$, an abstract string $O \in Strings$ is said to have* **property** $C = C(M, M')$ *if:*

$$\Theta_{M,M'} \implies \forall m : (*M'(M', m) \text{ does not halt}) \vee (M'(M', m) \notin \mathcal{O})$$
$$\vee\, (M'(M', m) \in \mathcal{O}, O \in \mathcal{O} \text{ and } X(\Sigma, m) = D_1 \circ R \circ \widetilde{M}'(M', m),\ \text{where } O = (X, \Sigma))$$

for $\widetilde{M}'$ determined by M' as in (2.8).

At a glance, this is a somewhat complicated property, but essentially it just says that if $\Theta_{M,M'}$ then for all m "$O \neq M'(M', m)$" unless either $*M'(M', m)$ does not halt, or does not have the right (data) type, or $R(O, m) = \infty$. The fact that data types get intricated is perhaps not surprising since there is a well known correspondence, the Curry-Howard correspondence, between proof theory and type theory in computer science.

Definition 2.10. *We say that $M \in \mathcal{M}_0$ is C-***sound** *if for each $(M', m) \in \mathcal{I}_0$, with $M(M', m) = O$ defined, O has property $C(M, M')$. We say that M is C-sound on M' if the list $\{M(M', m)\}_m$ has only elements with property $C(M, M')$.*

Thus $M \in \mathcal{M}_0$ is C-sound iff it prints strings with property C, which expresses a certain "diagonal" property with respect to the input. By "diagonal" we mean that this setup has something analogous to Cantor's diagonalization argument, but we will not elaborate.

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

Definition 2.11. *If M, M' as above are C-sound we will say that $sound(M)$, $sound(M')$ hold. If M is C-sound on M' we say that $sound(M, M')$ holds.*

Example 1. A trivially C-sound machine M is one for which

$$M(M', m) = (D_1 \circ R \circ \widetilde{M}', M')$$

for every $(M', m) \in \mathcal{I}_0$. As $(D_1 \circ R \circ \widetilde{M}', M')$ automatically has property $C(M, M')$ for each $M' \in \mathcal{T}_0$. In general, for any M, M' the list of all strings $\{(X, \Sigma)\}$ with property $C(M, M')$ is always infinite, as by this example there is at least one such string $(D_1 \circ R \circ \widetilde{M}', M')$, which can then be modified to produce infinitely many such strings.

Theorem 2.12. *If $sound(M, M') \wedge \Theta_{M,M'}$ then*

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

On the other hand, if $sound(M, M')$ then the string

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

has property $C(M, M')$. In particular if $sound(M)$ then $\mathcal{G}$ has property $C(M, M')$ for all 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', such that 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, all that is needed is $\mathcal{M} \in \mathcal{M}_0$ and is C-sound. So $\mathcal{G}$ is in a sense universal.

Proof. Suppose not and let M'_0 be such that $\Theta_{M,M'_0} \wedge sound(M, M'_0)$ and such that

$$M(M'_0, m_0) = \mathcal{G} \text{ for some } m_0. \tag{2.13}$$

Set $T = (M'_0, m_0)$ then we have that:

$$\begin{aligned}
&1 = D_1(\infty, m_0),\\
&D_1(\infty, m_0) = D_1 \circ R \circ \widetilde{M}'(T)), \text{ by } \mathit{sound}(M, M'), \text{ and by } * M'(T) \to \mathcal{G} \in \mathcal{O},\\
&D_1 \circ R \circ \widetilde{M}'(T) = D_1 \circ R(D_1, \infty, m_0) \quad \text{ by } M(T) = \mathcal{G}, \text{ and by } \Theta_{M,M'}\\
&D_1 \circ R(D_1, \infty, m_0) = 2 \quad \text{ by property } G \text{ of } R \text{ and by (2.2)}.\\
&1 = 2.
\end{aligned}$$

So we obtain a contradiction.

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

$$\forall m : \left(\mathit{sound}(M, M') \wedge (M'(T) \in \mathcal{O}) \wedge \Theta_{M,M'} \implies R(\widetilde{M}'(T)) = \infty\right), \tag{2.14}$$

where $T = (M', m)$. Suppose otherwise that for some m_0 and $T_0 = (M', m_0)$ we have:

$$\mathit{sound}(M, M') \wedge (*M'(T_0) \text{ halts}) \wedge (M'(T_0) \in \mathcal{O}) \wedge \Theta_{M,M'} \wedge (R(\widetilde{M}'(T_0)) \neq \infty). \tag{2.15}$$

Then since $M'(T_0) = (X, \Sigma)$, for some $(X, \Sigma) \in \mathcal{O}$ and since R is everywhere defined:

$$R(\widetilde{M}'(T_0)) = R(X, \Sigma, m_0) = X(\Sigma, m_0) = x \in \mathbb{Z}, \text{ for some } x,$$

by Property G of R and by $R(\widetilde{M}'(T_0)) \neq \infty$.

Then we get:

$$x = X(\Sigma, m_0) = D_1 \circ R \circ \widetilde{M}'(T_0) = D_1(x) = x + 1$$

by $\mathit{sound}(M, M')$, $\Theta_{M,M'}$ and by (2.2), so we get a contradiction and (4.7) 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 $\mathcal{I} = \mathcal{T} \times \mathbb{N}$. 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. For practical purposes S has in his environment a general purpose digital computer with arbitrarily, as necessary, expendable memory, (in other words a universal Turing machine).

We suppose that upon receiving any $T \in \mathcal{I}$, as a string in his computer, after possibly using his computer in some way, S instructs his computer to print after some indeterminate time a string $S(T)$. We are not actually assuming that $S(T)$ is defined on every T, (although this would likely be a safe assumption). So S also denotes a machine in our language, or a partially defined function:

$$S : \mathcal{I} \to \mathit{Strings},$$

Definition 3.1. *We say that S the human subject is* **computable** *if the corresponding machine S above is computable.*

3.1. Additional conditions.

We now consider a more specific S_0 of the type above, which additionally behaves in the following way. For any fixed $B \in \mathcal{T}_0$

$$\{S_0(B, m)\}_m$$

is the complete list of strings that S_0 asserts to have property $C(S_0, B)$. Of course we don't actually need 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 $C(S_0, B)$ will appear.

Also as in the Penrose argument we ask that S_0 asserts that he is fundamentally sound. Our human subjects are assumed to be idealized, so that all the "brain noise" issues are stripped out of them. In other words for our idealized humans fundamental soundness and soundness conditions are equivalent. We suppose then that S_0 asserts his soundness, which entails in this case that he asserts *sound*(S_0) for S_0 the above machine. For the reader that objects that this is not very explicit as the fundamental soundness condition is not very explicit, we promise to fix this in a following section, by reinterpreting fundamental soundness in terms of a completely explicit "stable soundness" condition. Thus, the following may be understood to be a preliminary result.

Theorem 3.2.

$$S_0 \textit{ is computable } \implies \neg \textit{sound}(S_0).$$

In fact we prove more, for any $S' \in \mathcal{T}_0$:

$$\Theta_{S_0,S'} \implies \neg \textit{sound}(S_0, S').$$

This partly formalizes Theorem 0.1.

Proof. Suppose $\Theta_{S_0,S'}$ for some $S' \in \mathcal{T}_0$. Suppose in addition *sound*(S_0, S'). Then by Theorem 2.12

$$S_0(S', m) \neq (D_1, \infty)$$

for any m. On the other hand S_0 asserts *sound*(S_0) and hence must assert that (D_1, ∞) has property $C(S_0, S')$, by the second half of Theorem 2.12. In particular the string (D_1, ∞) must be in the list $\{S_0(S', m)\}_m$, since this list is assumed to be complete. So we have reached a contradiction. ☐

4. Fundamental soundness as stable soundness

Imagine a machine M which sequentially prints statements of arithmetic, which it asserts are true, but so that M can also go back and delete a statement it later decided was untrue after all. We say that M is stably sound if any printed statement by M that survives to infinity is in fact true. Or in other words if whenever M prints a statement and never changes its mind on it, then this statement is true. More formally, for each $n \in \mathbb{N}$, $M(n)$ will correspond to an operation denoted by the string $(\Sigma, +)$ or $(\Sigma, -)$ meaning add Σ to the list or remove Σ from list, respectively, where Σ is a statement of arithmetic. Then we say that M is ***stably sound***: if whenever there is an n_0 with $M(n_0) = (\Sigma_0, +)$, s.t. there is no $m > n$ with $M(m) = (\Sigma_0, -)$, then Σ_0 is true.

We now specialize this to our setting. The crucial point of our Gödel string is that it will still function in this stable soundness context.

Let $\mathcal{M}^{\pm}$ denote the set of machines

$$M : \mathcal{I} \to \textit{Strings} \times \{\pm\},$$

where $\{\pm\}$ is the set containing two symbols $+, -$, implicitly encoded as a subset of *Strings*. We denote by $\mathcal{T}^{\pm}$ the subset corresponding to Turing machines. For each $M \in \mathcal{M}_0$, we define a machine:

$$\widetilde{M} : \mathcal{I} \to \textit{Strings} \times \mathbb{N}$$

$$\widetilde{M}(B, m) = (pr \circ M(B, m), m), \tag{4.1}$$

which is naturally a Turing machine when M is a Turing machine.

Definition 4.2. *For $M \in \mathcal{M}^{\pm}$, and for $T = (B, m) \in \mathcal{I}$ s.t. $M(T) = (O, +)$, we say that $M(T)$ is* **stable** *if there is no $k > m$ s.t. $M(B, k) = (O, -)$. In this case we also call O M*-**stable**.

In what follows $\mathcal{O} \subset \mathcal{T}_{\mathbb{Z}} \times \textit{Strings}$ is as before.

Definition 4.3. *For $M \in \mathcal{M}^{\pm}$, $M' \in \mathcal{T}^{\pm}$, an abstract string $O \in \textit{Strings}$ is said to have* **stable property** $C = C(M, M')$ *if:*

$$\Theta_{M,M'} \implies \forall m : (*M'(M', m) \textit{ does not halt}) \vee (pr \circ M'(M', m) \notin \mathcal{O}) \vee (M'(M', m) \textit{ is not stable})$$
$$\vee\, (pr \circ M'(M', m) \in \mathcal{O}, O \in \mathcal{O} \textit{ and } X(\Sigma, m) = D_1 \circ R \circ \widetilde{M}'(M', m), \textit{ where } O = (X, \Sigma)))$$

for $\widetilde{M}'$ determined by M' as in (4.1).

Definition 4.4. *We say that $M \in \mathcal{M}^{\pm}$ is* **stably** C**-sound** *on M', and we write that $s-sound(M, M')$ holds, if for each m with $M(M', m) = (X, \Sigma, +)$ stable, (X, Σ) has stable property $C(M, M')$. We say that M is* **stably** C**-sound** *if it is stably C-sound on all M', and in this case we write that $s - sound(M)$ holds.*

Example 2. As before an example of a trivially stably C-sound machine M is one for which

$$M(M', m) = (D_1 \circ R \circ \widetilde{M}', M', +)$$

for every $(M', m) \in \mathcal{I}_0$.

Theorem 4.5. *If $s - sound(M, M') \wedge \Theta_{M,M'}$ then*

$$\forall m \text{ s.t. } M(M', m) \text{ is stable} : M(M', m) \neq (D_1, \infty, +).$$

On the other hand, if $s - sound(M, M')$ then the string

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

has stable property $C(M, M')$. In particular if $s - sound(M)$ then $\mathcal{G}$ has stable property $C(M, M')$ for all M'.

Proof. This is mostly analogous to the proof of Theorem 2.12. Suppose not and let M' be such that $\Theta_{M,M'} \wedge sound(M, M')$ and such that for some m_0:

$$M(M', m_0) = (\mathcal{G}, +) \text{ and } \mathcal{G} \text{ is } M\text{-stable.} \tag{4.6}$$

To recall we have that:

$$1 = D_1(\infty, m_0).$$

If we set $T = (M', m_0)$, then also by $s - sound(M, M')$, by $*M'(T) \to (\mathcal{G}, +)$, $\mathcal{G} \in \mathcal{O}$ since $\Theta_{M,M'}$ and by $\mathcal{G}$ is M'-stable since $\Theta_{M,M'}$:

$$D_1(\infty, m_0) = D_1 \circ R \circ \widetilde{M}'(T).$$

On the other hand:

$$\begin{aligned}
&D_1 \circ R \circ \widetilde{M}'(T) = D_1 \circ R(D_1, \infty, m_0) \quad \text{ by } M(T) = (\mathcal{G}, +), \text{ and by } \Theta_{M,M'} \\
&D_1 \circ R(D_1, \infty, m_0) = 2 \quad \text{ by property } G \text{ of } R \text{ and by (2.2).} \\
&1 = 2.
\end{aligned}$$

So we obtain a contradiction.

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

$$s - sound(M, M') \wedge (M'(T) \in \mathcal{O}) \wedge (M'(T) \text{ is stable}) \wedge \Theta_{M,M'} \implies R(\widetilde{M}'(T) = \infty. \tag{4.7}$$

Suppose otherwise that for some m_0 and $T_0 = (M', m_0)$, we have:

(4.8)

$$s-sound(M, M') \wedge (*M'(T_0) \text{ halts}) \wedge (M'(T_0) \in \mathcal{O}) \wedge (M'(T_0) \text{ is stable}) \wedge \Theta_{M,M'} \wedge (R(\widetilde{M}'(T_0)) \neq \infty).$$

Then by this condition:

$$*M'(T_0) \to (X, \Sigma, +) \in \mathcal{O},$$

for some (X, Σ), M'-stable with property $C(M, M')$. Since R is everywhere defined:

$$R(\widetilde{M}'(T_0)) = R(X, \Sigma, m_0) = X(\Sigma, m_0) = x \in \mathbb{Z}, \text{ for some } x,$$

by Property G of R and by $R(\widetilde{M}'(T_0)) \neq \infty$. Then we have:

$$x = X(\Sigma, m_0) = D_1 \circ R \circ \widetilde{M}'(T_0) = x + 1$$

by $s - sound(M, M')$, by $\Theta_{M,M'}$, and by (2.2). So we get a contradiction and (4.7) follows. Our conclusion readily follows. ☐

5. A system with a human subject S as a machine in $\mathcal{M}^{\pm}$

Let S be in an isolated environment as before. We may then suppose as in Section 3 that S determines an element of $\mathcal{M}^{\pm}$:

$$S : \mathcal{I} \to \mathit{Strings} \times \{\pm\}.$$

Definition 5.1. *As before, we say that S the human subject is* **computable** *if the corresponding machine S above is computable.*

5.1. **Additional assumptions.** We now put the following additional assumptions. For any fixed $B \in \mathcal{T}^{\pm}$, if S ever asserts that (X, Σ) has stable property $C(S, B)$ then

$$(X, \Sigma, +) = S(B, m)$$

for some m, and conversely if $S(B, m) = (X, \Sigma, +)$ for some m then at some point S asserts that (X, Σ) has stable property $C(S, B)$. We also ask that S stably asserts that they are fundamentally sound, which in the specific setting here means that S stably asserts $s - \mathit{sound}(S)$. Here stably means unshakeably, meaning that S is never to change their mind on this.

Remark 5.2. It makes good sense now for S to stably assert $s - \mathit{sound}(S)$, at least for an idealized S whose brain is not subject to expiration. Given such an idealization, S is simply asserting (in the very limited context above) that the list of what they assert to be true converges (in the exact sense above) to a list of things actually true. For example I assert in absolute faith M: 5 is an odd number. This M is likely stably on my list, unless I would have lost my sanity and hence would no longer be me. If the reader does not like the idealization above, then they may replace S by "the indefinite scientific community" C. Applying the argument to this C yields an equivalent obstruction.

Theorem 5.3.

$$S \text{ is computable } \implies \neg s - \mathit{sound}(S).$$

That is if our subject S is computable they cannot be fundamentally sound, specifically meaning stably sound. In fact we prove more, for any $S' \in \mathcal{T}^{\pm}$:

$$\Theta_{S,S'} \implies \neg s - \mathit{sound}(S, S').$$

This formalizes Theorem 0.1.

Proof. Suppose $\Theta_{S,S'}$ for some $S' \in \mathcal{T}^{\pm}$. Suppose in addition $s - \mathit{sound}(S, S')$. Then by Theorem 4.5 for all m s.t. $S(S', m)$ is stable:

$$S(S', m) \neq (D_1, \infty, +).$$

On the other hand S stably asserts $s - \mathit{sound}(S)$ and hence must stably assert that (D_1, ∞) has stable property $C(S, S')$, by the second half of Theorem 4.5. In particular the string $(D_1, \infty, +)$ must be on the list $\{S(S', m)\}_m$, since this list is assumed to be complete, and moreover (D_1, ∞) is S-stable since by assumption S asserts $s - \mathit{sound}(S)$ stably. So we have reached a contradiction. □

5.2. **Formal system interpretation.** This is not necessary for our main theorem, and is more technical, but in practice it might be helpful to interpret the above in terms of formal systems. For simplicity we will base everything of standard set theory $\mathcal{ST}$. Turing machines are assumed to be naturally formalized in $\mathcal{ST}$. In what follows, $\mathcal{F} \vdash P$ means that P is provable in $\mathcal{F}$.

Definition 5.4. *We will say that $S \in \mathcal{M}^{\pm}$, the machine as above,* **is captured by a formal system** *$\mathcal{F} \supset \mathcal{ST}$ if:*

$$(\exists m : S(S', m) = (X, \Sigma, +) \text{ is stable}) \iff (\mathcal{F} \vdash ((X, \Sigma) \text{ has stable property } C(S, S'))).$$

Note that $\mathcal{F}$ is not uniquely determined by this condition.

Let $Con(S)$ denote the meta-statement:

$$\exists \mathcal{F} : (\mathcal{F} \supset \mathcal{ST}) \wedge (\mathcal{F} \text{ captures } S) \wedge (\mathcal{F} \text{ is consistent}).$$

In what follows by "provably" we mean provably in $\mathcal{ST}$.

Theorem 5.5. *Let S be as above then:*

$$(\exists S' \in \mathcal{T}^{\pm} \text{ so that provably } \Theta_{S,S'}) \implies \neg Con(S),$$

or in more logic symbols,

$$(\exists S' \in \mathcal{T}^{\pm} : \mathcal{ST} \vdash \Theta_{S,S'}) \implies \neg Con(S).$$

In particular if provably $\Theta_{S,S'}$, for some S', and if S is captured by $\mathcal{F} \supset \mathcal{ST}$ then

$$\{pr \circ S(S', m) | m \in \mathbb{N}, \text{ s.t. } S(S', m) \text{ is stable}\} = Strings.$$

Note that "provably $\Theta_{S,S'}$" does not mean that S can prove $\Theta_{S,S'}$ in the practical sense. It just means that after the terms S, S' in the statement $\Theta_{S,S'}$ have been completely interpreted in set theory $\mathcal{ST}$, $\Theta_{S,S'}$ is provable in $\mathcal{ST}$. But interpretation of the term S may not even be practically attainable by S, as S is underlaid by some very complex physical system. And even if this interpretation was attainable, S may not be clever enough to find the proof of $\Theta_{S,S'}$, again in the practical sense. Also note that $\neg Con(S)$ expresses *fundamental* inconsistency of S, as we only take stable assertions of S above. The second part of the theorem expresses a bizarre consequence that S stably (unshakeably) asserts that even completely non-sense strings have stable property $C(S, S')$.

Proof. Let $\mathcal{F}(S)$ capture S as above, and $S' \in \mathcal{T}^{\pm}$. By the second part of Theorem 5.3,

$$\mathcal{ST} \vdash (\Theta_{S,S'} \implies L),$$

where $L = L(S, S')$ is:

$$\exists m : (S(S', m) \text{ is defined and is stable}) \wedge (S(S', m) \text{ does not have property } C(S, S')).$$

So if provably $\Theta_{S,S'}$, L is provable in $\mathcal{ST}$ and hence in $\mathcal{F}(S)$. On the other hand, by assumption that S is captured by $\mathcal{F}(S)$, $\neg L$ is provable in $\mathcal{F}(S)$. Then the first part of the theorem follows. The second part is then immediate, since if $\mathcal{F}(S)$ is inconsistent it proves that every string has stable property $C(S, S')$, it proves everything in fact. □

6. Concluding remark

While it is not hard to argue that humans are simply not sound, based on empirical evidence alone, it would be much more difficult to argue that we are not stably sound. After all the entire body of science would be meaningless if we were. Thus our results put a very serious obstruction to computability of intelligence.

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

References

[1] A.M. Turing. "Computing machines and intelligence". In: *Mind* 49 (1950), pp. 433–460.
[2] A.M. Turing. "On computable numbers, with an application to the entscheidungsproblem ". In: *Proceedings of the London mathematical society* s2-42 (1937).
[3] K. Gödel. *Collected Works III (ed. S. Feferman)*. New York: Oxford University Press, 1995.
[4] S. Feferman. "Are There Absolutely Unsolvable Problems? Gödel's Dichotomy". In: *Philosophia Mathematica* 14.2 (2006), pp. 134–152.
[5] J.R. Lucas. "Minds machines and Gödel". 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] Roger Penrose. “Beyond the shadow of a doubt”. In: *Psyche* (1996). URL: http:%5C%5Cpsyche.cs.monash.edu.au%5Cv2%5Cpsyche-2-23-penrose.html.

[15] 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.

[16] 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.

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

University of Colima, CUICBAS
Email address: yasha.savelyev@gmail.com