Direct link to author's version

UNIVERSAL 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. To this end, we develop in a specific setting an analogue of a Gödel statement for stably sound Turing machines, as defined here. This stably Gödel statement $\mathcal{G}$ in addition has a certain universality with respect to a certain class of Turing machines / formal systems. Moreover, this $\mathcal{G}$ is constructed explicitly, given the general form of our class of Turing machines. The main argument is implicitly based on set theory, in contrast to the partially meta-logical arguments of Gödel and Penrose, arguing for a similar obstruction. For this reason we are able to phrase our results as theorems.

In what follows we understand *human intelligence* very much like Turing in [2], as 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 - the inner workings of human intelligence that are supposed as special. The only thing that concerns us is what output B produces given an input, not 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 [1]) 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, defined in a certain precise sense. For a non-expert reader we point out that this "fixed" does not preclude the algorithm from "learning", [1] it just means that how it "learns" is completely determined by the initial 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 some particular program. Unbounded memory is just a mathematical convenience. In specific arguments, also of the kind we make, we can work with 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 have any relevance we need some assumption on the soundness or consistency of human reasoning. Informally, a human is sound if whenever they 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. However, we cannot honestly hope for soundness 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). But we can certainly hope for some kind of fundamental soundness.

In this work we will formally interpret fundamental soundness as stable soundness. Essentially, our machine [2] B is now allowed to make corrections, and if a statement printed by B is never corrected then this statement is true, if B has our stable soundness property. The negation of stably sound is stated as either stably unsound or not stably sound, synonymously. This stable soundness reflects our basic understanding of how science progresses. Of course even stable soundness needs idealizations to

[1] In the sense of "machine learning".

[2] Here we use the term machine as an abstraction for a process acting on inputs, but it need not be a computational process, in contrast to Turing machines.

make sense for humans. The human brain deteriorates and eventually fails, so that either we idealize the human brain to never deteriorate, or B now refers not to an individual human but to the evolving scientific community.

Around the same time as Turing, Gödel argued for a no answer to Question 1, see [13, 310], relating the question to existence of absolutely undecidable problems, see also Feferman [8], and Koellner [15], [16] 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. Interestingly, for Gödel fundamental unsoundness of human reasoning is not even a possibility, it does not seem to be stated in [13]. A more in depth analysis of Gödel versus Turing on computability and the mind appears for example in [5].

Later Lucas [12] and later again and more robustly Penrose [20] argued for a no answer based only on soundness, and by further elaborating the obstruction from the Gödel incompleteness theorem. Such an argument if correct would be much more compelling.

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 [11] 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: [14], [17], [9], [10].

From my perspective, the most troubling issue with the Penrose argument concerns the soundness assumption, and we will review this in this introduction. Other authors, particularly Koellner [15], [16], argue that there are meta-logical issues even allowing for soundness. Following a technically very different and much more elementary approach, in relation to Penrose, we intend to completely resolve these issues here. In particular, there should be no meta-logical issues because our results are theorems and not just meta-theorems, cf. Section 6.

The following is a slightly informal version of our main Theorem 4.5, taking the view that our idealized human is represented by the evolving scientific community or just H for short.

Theorem 0.1. *Either there are cognitively meaningful, absolutely non Turing computable processes in the human brain or human beings are fundamentally unsound, meaning specifically that H is stably unsound. This theorem is indeed a mathematical fact,* [3] *given our formalization of H and of stable soundness.*

By *absolutely* we mean in any physical model. Note that even existence of absolutely 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. [3]) 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. It would be good to compare this theorem this with Deutch [7], where computability of any suitably finite and discreet physical system is conjectured. Although this is not immediately at odds with us, as the hypothesis of that conjecture may certainly not be satisfiable.

By strengthening the hypothesis of Theorem 0.1 from computability to provable computability as in Theorem 5.2, we can obtain more practical consequences. To the effect that not only is H stably unsound but must in fact eventually stably assert $0 = 1$.

0.1. **The Penrose argument.** Following Lucas [12], Penrose has given variations of the argument for a no answer to Question 1 in his books [18], [19]. The final argument can be found in [20], which we now re-interpret in a language closer to our subsequent argument. The following argument (really just an outline) is more elaborate then what was originally proposed by Penrose, but this is because we make some additional things explicit.

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.

[3] Specifically a theorem of set theory, although we keep set theory implicit as usual.

Suppose a human subject 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 $n \in \mathbb{N}$. The output $P(\Sigma_T, n)$ is a statement of arithmetic printed by P. (For now there is no requirement on truth.)

Given a Turing machine T let Θ_T be the statement:

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

We ask that for each fixed T: $\{P(\Sigma_T, n)\}_n$ is the complete list of statements that P perceives [4] to be true conditionally on Θ_T. This may look somewhat self referential, but this can be fixed, cf. Section 4. It follows by the above that P must also perceive for each T the statement I_T:

$$\Theta_T \implies T \text{ is sound}, \tag{0.3}$$

which means that $T(\Sigma_T, n)$ is true for each n.

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 [5], the statements $\{T_0(\Sigma_{T_0}, n)\}_n$ [6] must be the complete list of provable statements in a certain formal system $\mathcal{F}(T_0)$, explicitly constructible given T_0. 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.

By construction $\mathcal{F}(T_0)$ would be sound if Θ_{T_0} and if I_{T_0}. In particular if $\Theta_{T_0} \wedge I_{T_0}$ then by the celebrated Gödel incompleteness theorem there would be a true Gödel statement $G(T_0)$ for this $\mathcal{F}(T_0)$, such that

$$T_0(\Sigma_{T_0}, n) \neq G(T_0), \quad \text{for all } n.$$

But P perceives I_{T_0}, hence he must perceive by implication that

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

And so assuming P knows 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 we conclude that either not Θ_{T_0}, that is P is not computed by T_0 or P is not sound, but T_0 is arbitrary so we obtain an obstruction to computability of P.

Even if it was in essence correct, the above argument is unsatisfactory because all it claims to prove is: either we are non-computable or unsound, which we appear to be anyway. Of course as we have argued we must talk of fundamental soundness, interpreted here as stable soundness. But then the argument cannot work exactly as above, since Gödel's theorem necessitates total soundness. This is a highly non-trivial issue, as while we can abstractly extract from a stably sound machine an absolutely sound machine, the latter may not be computable even if the former was, cf. Remark 3.2. So we cannot just hope to fix things by saying our human is idealized without being extremely detailed what "idealized" means physically, for if idealized just means stabilized as in Remark 3.2 then such an idealization may introduce non-computability.

We will delve no further into critiquing the Lucas-Penrose argument. One such critique is given in Koellner [15], [16], see also Penrose [20], and Chalmers [4] for discussions of some issues. Note of course that our version of the Penrose argument is slightly different, and so the issues might be different.

So motivated by the discussion above, the ideal thing to do is to formally define stable 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. To this end, we reformulate the above idea using a more elementary approach, more heavily based in Turing machines. We first isolate a certain class of Turing machines that we name diagonalization machines. They print strings with a certain property C. As the name suggests, their behavior is related to the Cantor diagonalization argument. Next we explicitly construct a "Gödel string" $\mathcal{G}$ which is universal for this whole class. This string $\mathcal{G}$ has property C but cannot be printed by a Turing diagonalization machine. Crucially, this is then

[4] "Perceive" can be understood to mean "assert to be true with absolute faith".

[5] I don't know a standard reference but see for example [8].

[6] Strictly speaking after taking deductive closure.

extended to stable diagonalization machines, which print property C [7] strings only stably. Given this, our main result follows by an argument similar to the one in the outline above.

This is essentially as far as we can go in trying to outline the argument, as most of it just concerns the construction of the class of diagonalization machines and of $\mathcal{G}$, and this is hard to describe without details. However, 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 Γ. A special symbol $b \in \Gamma$ 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 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 actions taken, 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.*

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. Given a total state s, $\delta(s)$ will denote the successor state of s, obtained by executing the instructions set of M on s, or in other words $\delta(s)$ is one step forward from s.*

So a Turing machine determines a special kind of function:

$$\delta^M : \mathcal{C}(M) \to \mathcal{C}(M),$$

where $\mathcal{C}(M)$ is the set of possible total states of M.

Definition 1.3. *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, and where $s_{i+1} = \delta(s_i)$. If elements of $\{s_i\}_{i=0}^{i=\infty}$ are eventually in some final machine state,

[7] The property is not exactly the same, it has to be suitably stabilized.

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 will 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 1.4. *Let Strings denote the set of all finite strings, including the empty string ϵ, of symbols in some fixed finite alphabet, with at least 2 elements. Given a partial 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*

$$*M(\Sigma) \to f(\Sigma) \text{ whenever } f(\Sigma) \text{ is defined.}$$

So a Turing machine T itself determines a partial 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 1.5. *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 will say that Σ is T-***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***.

We will sometimes use abstract sets to refer to input and output sets. However, these are understood to be subsets of *Strings* under some implicit, *fixed* encoding. Concretely an ***encoding*** of A is an injective set map $i : A \to Strings$. For example if the input set is $Strings^2$, we may encode it as a subset of *Strings* as follows. The encoding string of $\Sigma \in Strings^2$ will be of the type: "this string encodes an element $Strings^2$, whose components are Σ_1 and Σ_2." In particular the sets of integers $\mathbb{N}, \mathbb{Z}$, which we use often, will under some encoding correspond to subsets of *Strings*. Indeed this abstracting of sets from their encoding in *Strings* is partly what computer languages do. The fixing of the encoding can be understood as fixing the computer language.

The above will allow us to work with a set $\mathcal{T}$ of Turing machines, with abstract sets of inputs and outputs implicitly encoded as subsets of *Strings* as above. Note that $\mathcal{T}$ itself has an induced encoding, called its program. Of course, concretely $\mathcal{T}$ is nothing more then the set of Turing machines, with a distinguished final state called q_{reject}.

Definition 1.6. *We say that a Turing machine T computes a partial 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 Turing machine which rejects all input. Let us not elaborate further.

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:

$$(T^1_i T^2_i), (T^1_c T^2_c), (T^1_o T^2_o),$$

three heads H_i, H_c, H_o which have component heads H^j_i, H^j_c, H^j_o, $j = 1, 2$. It has machine states:

$$Q_{M^1 \star M^2} = Q^1 \times Q^2 \times (\mathbb{Z}_2 = \{0, 1\}),$$

with initial state $(q^1_0, q^2_0, 0)$ and final states:

$$F_{M^1 \star M^2} = F^1 \times Q^2 \times \{1\} \sqcup Q^1 \times F^2 \times \{0\}.$$

Clearly we have a natural splitting

$$\mathcal{C}(M^1 \star M^2) = \mathcal{C}(M^1) \times \mathcal{C}(M_2) \times \mathbb{Z}_2.$$

In terms of this splitting we define the transition function

$$\delta^{M^1 \star M^2} : \mathcal{C}(M^1 \star M^2) \to \mathcal{C}(M^1 \star M^2),$$

for our Turing machine $M^1 \star M^2$ by:

$$\delta^{M^1 \star M^2}(s^1, s^2, 0) = (\delta^{M^1}(s_1), s^2, 1)),$$
$$\delta^{M^1 \star M^2}(s^1, s^2, 1) = (s_1, \delta^{M^2}(s^2), 0)).$$

Or, concretely this means the following. Given machine state $q = (q^1, q^2, 0)$ and the symbols

$$(\sigma^1_i \sigma^2_i), (\sigma^1_c \sigma^2_c), (\sigma^1_o \sigma^2_o)$$

to which the heads H_i, H_c, H_o are currently pointing, we first check instructions in I^1 for q^1, $\sigma^1_i, \sigma^1_c, \sigma^1_o$, and given those instructions as step 1 execute:

(1) Replace symbols σ^1_c, σ^1_o to which the head components H^1_c, H^1_o point, or leave them unchanged, while leaving unchanged the symbols to which H^2_c, H^2_o point.
(2) Move each head component H^1_i, H^1_c, H^1_o left, right, or leave it in place, (independently). (The second components of the heads are unchanged.)
(3) Change the first component of q to another machine state in Q^1 or keep it, based on the instruction in I^1. Leave the second component of q unchanged. The third component of q is changed to 1.

Then likewise given machine state $q = (q^1, q^2, 1)$, we check instructions in I^2 for q^2, $\sigma^2_i, \sigma^2_c, \sigma^2_o$ and given those instructions as step 2 execute:

(1) Replace symbols σ^2_c, σ^2_o to which the head components H^2_c, H^2_o point, or leave them unchanged, while leaving unchanged the symbols to which H^1_c, H^1_o point.
(2) Move each head component H^2_i, H^2_c, H^2_o left, right, or leave it in place.
(3) Change the second component of q to another or keep it, based on instruction in I^2. Leave the first component unchanged, and change the third component of q to 0.

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.

1.2. **Universal Turing machines.** It will be convenient to refer to the universal Turing machine

$$U : \mathcal{T} \times \mathit{Strings} \to \mathit{Strings},$$

for $\mathcal{T}$ the set of Turing machines as already indicated above. This universal Turing machine already appears in Turing's [1]. It permits 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 string Σ 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 sometimes 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. Preliminary setup for the proof of Theorem 0.1

This section can be understood to be a warm up, as we will not yet work with stable soundness. But most of this will carry on to the more technical setup of Section 3.

Definition 2.1. *A* **machine** [8] *will be a synonym for a partial function $A : I \to O$, with I, O abstract sets with a fixed, 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 $fog(T)$ by forgetting all structure except the structure of a partial function. $\mathcal{T}$ will denote the set of machines, which in addition have the structure of a Turing machine. So we have a forgetful map $fog : \mathcal{T} \to \mathcal{M}$.

2.1. **Diagonalization machines.** There is a well known connection between Turing machines and formal systems, to which we already alluded in Section 0.1. So Gödel statements can already be interpreted in Turing machine language as certain Gödel strings. But we will be aiming to construct, in a specific setting relevant to our goals, a more flexible and in a certain sense universal (for our class of Turing machines) such Gödel string $\mathcal{G}$. Extending this construction to more general classes of Turing machines / formal systems would be very interesting, but at the moment it is not clear what that would entail.

To make this $\mathcal{G}$ exceptionally simple we will need to formulate some specific properties for our machines, which will require a bit of setup. We denote by $\mathcal{T}_{\mathbb{Z}} \subset \mathcal{T}$ the subset of Turing machines of the type:

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

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

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

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

[8] For some authors and in some of the writing of Turing and Gödel "machine" is synonymous with Turing machine. For us the term machine is just abstraction for a process.

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 element ∞, which is just a particular distinguished symbol, also implicitly encoded as an element of *Strings*, s.t. $\{\infty\} \cap \mathbb{Z} = \emptyset$, where the intersection is taken in *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*

$$R : D \supset \mathcal{O}' \to \mathbb{Z} \sqcup \{\infty\},$$

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

- *R halts on the entire $\mathcal{O}'$, that is $\mathcal{O}'$ is contained in the set of R-acceptable strings.*
- *$R(X, \Sigma, m) \neq \infty \implies R(X, \Sigma, m) = X(\Sigma, m)$, for $(\Sigma, m) \in S_X \times \mathbb{N}$, 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 the previous property.*

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

Proof. Let W_n be some Turing machine $W_n : \{\epsilon\} \to \{\infty\}$, for $\epsilon \in$ *Strings* 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 $*W_n(\epsilon)$ is $n > 0$, (cf. Preliminaries). Let R_n be the Turing machine specified as:

$$R_n(Z) := W_n \star U(\epsilon, Z),$$

in the language of the join operation described in Section 1, for $Z \in$ *Strings*, and for U the universal Turing machine. Clearly R_n always halts, although it may halt with machine state q_{reject}. Moreover by construction every $Z = (X, \Sigma, m) \in \mathcal{O}' \subset$ *Strings* is permitted. Additionally, for $(X, \Sigma, m) \in \mathcal{O}'$,

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

in particular every $(X, \Sigma, m) \in \mathcal{O}'$ is R_n-acceptable. 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 $n_0 > l$ is fixed. Then by construction we get:

$$\forall m : R_{n_0}(D_1, \infty, m) = D_1(\infty, m) = 1.$$

So set $R := R_{n_0}$, and this gives the desired Turing machine. Note that the domain $D \subset \mathcal{T} \times$ *Strings* of R-permissible strings is not explicitly determined by our construction, as we cannot tell without additional information when a general Z is rejected by R. We can only say that $D \supset \mathcal{O}'$. □

Define $\mathcal{M}_0$ to be the set of machines whose input set is $\mathcal{I} = \mathcal{T} \times \mathbb{N}$ and whose output set is *Strings*. That is

$$\mathcal{M}_0 := \{M \in \mathcal{M} | M : \mathcal{T} \times \mathbb{N} \to \textit{Strings}\}.$$

We set

$$\mathcal{T}_0 := \{T \in \mathcal{T} | fog(T) \in \mathcal{M}_0\},$$

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 $T(T, m)$, we mean $T(\Sigma_T, m)$ for Σ_T the string encoding of the specification of the Turing machine T. So we conflate the notation for the Turing machine and its string specification, i.e. program.

Definition 2.9. *For $M \in \mathcal{M}_0$, $T \in \mathcal{T}_0$, an abstract string $O \in Strings$ is said to have* **property** $C = C(M, T)$ *if:*

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

where $(X, \Sigma) = O$ and where $\widetilde{T}$ is determined by T as in (2.8).

At a glance, this is a somewhat complicated property, but essentially it just says that if $\Theta_{M,T}$ then for all m "$O \neq T(T, m)$" unless either $*T(T, m)$ does not halt, or the output does not have the right (data) type, or $R(O, m) = \infty$. Thus the string O with property $C(M, T)$ is "diagonal" in a certain sense, where by "diagonal" we mean that something analogous to Cantor's diagonalization is happening, but we will not elaborate.

Remark 2.10. *The fact that data types get intricated is perhaps not surprising. On one hand there is a well known correspondence, the Curry-Howard correspondence [6], between proof theory in logic and type theory in computer science, and on the other hand we are doing something at least loosely related to Gödel incompleteness, but in the language of Turing machines.*

Definition 2.11. *We say that $M \in \mathcal{M}_0$ is C-***sound***, or is a* **diagonalization machine***, if for each $(T, m) \in \mathcal{I}_0$, with $M(T, m) = O$ defined, O has property $C(M, T)$. We say that M is C-sound on T if the list $\{M(T, m)\}_m$ has only elements with property $C(M, T)$.*

Define a C-sound $T \in \mathcal{T}_0$ analogously.

Definition 2.12. *If M as above is C-sound we will say that sound(M) holds. If M is C-sound on T we say that sound(M, T) 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 $(M', m) \in \mathcal{I}$. 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 $M \in \mathcal{M}_0$, $M' \in \mathcal{T}_0$ the list of all strings O 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 2.13. *Given any $M \in \mathcal{M}_0$, if sound$(M, M') \wedge \Theta_{M,M'}$ for some $M' \in \mathcal{T}_0$ then*

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

where $\mathcal{G} := (D_1, \infty)$. On the other hand:

$$\forall T \in \mathcal{T}_0 : sound(T, T) \implies \mathcal{G} \text{ has property } C(T, T).$$

In particular if sound(M) then $\mathcal{G}$ has property $C(M, T)$ for all $T \in \mathcal{T}_0$.

In the second half of the above theorem we may treat an element $T \in \mathcal{T}_0$ as an element of $\mathcal{M}_0$ via the map fog. So given any C-sound $M \in \mathcal{M}_0$ there is a certain string $\mathcal{G}$ with property $C(M, T)$ for all $T \in \mathcal{T}_0$, such that for each $M' \in \mathcal{T}_0$ 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

$$M(M_0', m_0) = \mathcal{G} \text{ for some } m_0,$$

so that $\mathcal{G}$ has property $C(M, M_0')$. Set $I_0 := (M_0', m_0)$ then we have that:

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

$D_1(\infty, m_0) = D_1 \circ R \circ \widetilde{M}'(I_0)$, by $\mathcal{G}$ having property $C(M, M')$, and by $* M'(I_0) \to \mathcal{G} \in \mathcal{O}$ since $\Theta_{M,M'}$,

$D_1 \circ R \circ \widetilde{M}'(I_0) = D_1 \circ R(D_1, \infty, m_0)$ by $M'(I_0) = \mathcal{G}$,

$D_1 \circ R(D_1, \infty, m_0) = 2$ by property G of R and by (2.2),

$1 = 2$.

So we obtain a contradiction.

We now verify the second part of the theorem. We show that:

$$\forall m, \forall T \in \mathcal{T}_0 : \left(sound(T, T) \wedge (T(T, m) \in \mathcal{O}) \implies R(\widetilde{T}(T, m)) = \infty \right). \tag{2.14}$$

Suppose otherwise that for some m_0, T_0 and $I_0 := (T_0, m_0)$ we have:

$$sound(T_0, T_0) \wedge (*T_0(I_0) \text{ halts}) \wedge (T_0(I_0) \in \mathcal{O}) \wedge (R(\widetilde{T}_0(I_0)) \neq \infty).$$

So we have:

$$* T_0(I_0) \to (X, \Sigma) \in \mathcal{O}, \tag{2.15}$$

for some (X, Σ) having property $C(T_0, T_0)$, by $sound(T_0, T_0)$. And so, since R is defined on all of $\mathcal{O}'$:

$$R(\widetilde{T}_0(I_0)) = R(X, \Sigma, m_0) = X(\Sigma, m_0) = x \in \mathbb{Z}, \text{ for some } x,$$

by Property G of R and by $R(\widetilde{T}_0(I_0)) \neq \infty$.

Then we get:

$$x = X(\Sigma, m_0) = D_1 \circ R \circ \widetilde{T}_0(I_0) = D_1(x) = x + 1$$

by (X, Σ) having property $C(T_0, T_0)$, and by (2.15). So we get a contradiction and (2.14) follows. Our conclusion readily follows.

The last part of the theorem follows by logic, for we have:

$$\forall T \in \mathcal{T}_0 : sound(T, T) \implies \mathcal{G} \text{ has property } C(T, T).$$

Also

$$\forall T : \Theta_{M,T} \wedge sound(M) \implies sound(T, T),$$

therefore

$$\forall T : sound(M) \implies (\Theta_{M,T} \implies \mathcal{G} \text{ has property } C(T, T)).$$

Which is the same as:

$$\forall T : sound(M) \implies \mathcal{G} \text{ has property } C(M, T),$$

by the definition of property $C(M, T)$.

□

2.2. **A system with a human subject S as a machine in $\mathcal{M}_0$.** Let S be a human subject, in a controlled environment, in communication with an experimenter/operator E that as input passes to S elements of $\mathcal{I} = \mathcal{T} \times \mathbb{N}$. Here ***controlled 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. This condition is only for simplicity, so long as we know in principle, or can compute in principle, what "input" our S receives, it doesn't matter what kind of environment he is in. 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 $I \in \mathcal{I}$, 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 $S(I)$. We are not actually assuming that $S(I)$ is defined on every I. So S in our language also denotes a machine:

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

which we suppose satisfies the condition that for any fixed $T \in \mathcal{T}_0$

$$\{S(T, m)\}_m$$

is the complete list of strings O, such that the physical S asserts:

$$\Theta_{S,T} \implies O \text{ has property } C(T, T).$$

In other words this is the list of strings that the physical S asserts to have property $C(S, T)$. (While being a part of the system above.) The reader may worry that the above is characterization of S is self-referential. In Section 4 we show how to fix this. For now, since this discussion is preliminary, we ignore the issue.

Remark 2.16. *The above is partially a simplification, because for a real world S it may be that each $S(T, m)$ must be understood as a probability distribution on Strings. In other words the value $S(T, m)$ is only determined up to some dice roll, which we may expect if quantum mechanics plays a significant role. This extra complexity will be ignored, as it does not meaningfully change any of our arguments, since dice rolls can be simulated completely with Turing machines.*

Definition 2.17. *We say that S the human subject is* **computable** *if the corresponding machine S above is computable.*

Penrose property. By the above conditions S must also assert the statement

$$\forall T \in \mathcal{T}_0 : \Theta_{S,T} \implies sound(T)$$

for S the above machine. And we ask that S is aware of Theorem 2.13, so that as a consequence S asserts that

$$\forall T \in \mathcal{T}_0 : \Theta_{S,T} \implies \mathcal{G} \text{ has property } C(T, T),$$

which is the same as

$$\forall T \in \mathcal{T}_0 : \mathcal{G} \text{ has property } C(S, T),$$

by the definition of property $C(S, T)$. Again by our condition on S, we may state this more concretely as:

$$\forall T \in \mathcal{T}_0 : \mathcal{G} = S(T, m) \text{ for some } m.$$

Theorem 2.18.

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

In fact we prove more, for any $T \in \mathcal{T}_0$:

$$\Theta_{S,T} \implies \neg sound(S, T).$$

This partly formalizes Theorem 0.1, to completely formalize it we must wait till the following sections.

Proof. Suppose $\Theta_{S,S'}$ for some $S' \in \mathcal{T}_0$. Suppose in addition $sound(S, S')$. Then by Theorem 2.13

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

for any m, which contradicts the Penrose property above. □

3. Fundamental soundness as stable soundness

Imagine a machine P which sequentially prints statements of arithmetic, which it asserts are true, but so that P can also delete a printed statement, if P decided the statement to be untrue. We say that P is stably sound if any printed statement by P that survives to infinity is in fact true. More formally, for each $n \in \mathbb{N}$, $P(n)$ 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. If there is an n_0 with $P(n_0) = (\Sigma, +)$, s.t. there is no $m > n_0$ with $P(m) = (\Sigma, -)$, then Σ is called ***P-stable*** and we say that P ***prints*** Σ ***stably***.

Definition 3.1. *We say that P is* **stably sound** *if every P-stable Σ is true.*

Remark 3.2. *Given a stably sound P, we may construct from it a sound machine P^s simply by enumerating, in order, all the P-stable Σ. However this is of no usefulness in our context, as in general P^s may not be computable even if P is computable. Explicit examples of this sort can be constructed by hand. We can in fact construct a Turing machine A, whose stabilization A^s enumerates every Diophantine equation with no integer solution, and this set is well known to be not computably enumerable,* [1].

We now translate 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

$$M : \mathcal{I} = \mathcal{T} \times \mathbb{N} \to Strings \times \{\pm\},$$

where $\{\pm\}$ is the set containing two symbols $+, -$, likewise implicitly encoded as a subset of $Strings$. We set

$$\mathcal{T}^{\pm} := \{T \in \mathcal{T} | fog(T) \in \mathcal{M}^{\pm}\}.$$

Definition 3.3. *For $M \in \mathcal{M}^{\pm}$, and for $(T, m) \in \mathcal{I}$, we say that an abstract $O \in Strings$ is (M, T)-***stable***, and that M* **prints** *O T-***stably*** *if there exists an $m \in \mathbb{N}$ s.t. $M(T, m) = (O, +)$ and there is no $k > m$ s.t. $M(T, k) = (O, -)$. When $T \in \mathcal{T}^{\pm}$ and $fog(T) = M$, instead of writing (M, T)-stable we just write T-stable.*

Let

$$pr : Strings \times \{\pm\} \to Strings,$$

be the natural projection. For each $M \in \mathcal{M}^{\pm}$, we define a machine:

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

$$\widetilde{M}(T, m) = (pr \circ M(T, m), m), \tag{3.4}$$

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

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

Definition 3.5. *For $M \in \mathcal{M}^{\pm}$, $M' \in \mathcal{T}^{\pm}$, an abstract string $O \in Strings$ is said to have property $sC = sC(M, M')$ if:*

$\Theta_{M,M'} \implies \forall m : (*M'(M', m)$ *does not halt*$) \vee (pr \circ M'(M', m) \notin \mathcal{O}) \vee (pr \circ M'(M', m)$ *is not M'-stable*$)$

$\vee (pr \circ M'(M', m) \in \mathcal{O}, O \in \mathcal{O}$ *and* $X(\Sigma, m) = D_1 \circ R \circ \widetilde{M'}(M', m),$ *where* $(X, \Sigma) = O)),$

for $\widetilde{M'}$ determined by M' as in (3.4).

Definition 3.6. *We say that $M \in \mathcal{M}^{\pm}$ is* **stably C-sound** *on M', and we write that $s-sound(M, M')$ holds, if every (M, M')-stable O has property $sC(M, M')$. 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

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

for every $(T,m) \in \mathcal{I}$.

Theorem 3.7. *For all $M \in \mathcal{M}^{\pm}$:*

$$(\exists M' \in \mathcal{T}^{\pm} : s-sound(M,M') \wedge \Theta_{M,M'}) \implies ((O \text{ is } (M,M')\text{-stable}) \implies O \neq \mathcal{G})$$

where

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

On the other hand,

$$\forall T \in \mathcal{T}^{\pm} : s-sound(T,T) \implies \mathcal{G} \text{ has property } sC(T,T). \tag{3.8}$$

Proof. This is mostly analogous to the proof of Theorem 2.13. Suppose not, let M be fixed and let M_0' be such that $s-sound(M,M_0') \wedge \Theta_{M,M_0'}$ and such that for some m_0:

$$M(M_0', m_0) = (\mathcal{G}, +) \text{ and } \mathcal{G} \text{ is } (M, M_0')\text{-stable},$$

with $\mathcal{G}$ consequently having property $sC(M,M_0')$, by $s-sound(M,M_0')$.

If we set $I_0 := (M_0', m_0)$, then by $\mathcal{G}$ having property $sC(M,M_0')$, by $*M_0'(I_0) \to (\mathcal{G},+)$, $\mathcal{G} \in \mathcal{O}$ since $\Theta_{M,M_0'}$ and by $\mathcal{G}$ being M_0'-stable as $\mathcal{G}$ is (M,M_0')-stable:

$$D_1(\infty, m_0) = D_1 \circ R \circ \widetilde{M_0'}(I_0). \tag{3.9}$$

On the other hand:

$$D_1 \circ R \circ \widetilde{M_0'}(I_0) = D_1 \circ R(D_1, \infty, m_0) \quad \text{by } M_0'(I_0) = (\mathcal{G}, +), \tag{3.10}$$

$$D_1 \circ R(D_1, \infty, m_0) = 2 \quad \text{by property } G \text{ of } R \text{ and by (2.2)}, \tag{3.11}$$

$$D_1(\infty, m_0) = 1, \tag{3.12}$$

$$1 = 2, \text{by (3.9) and by (3.11)}. \tag{3.13}$$

So we obtain a contradiction.

We now verify the second part of the theorem. Given any $T \in \mathcal{T}^{\pm}$, for any $m \in \mathbb{N}$, setting $I := (T,m)$ we show that:

$$s-sound(T,T) \wedge (pr \circ T(I) \in \mathcal{O}) \wedge (pr \circ T(I) \text{ is } T\text{-stable}) \implies R(\widetilde{T}(I)) = \infty. \tag{3.14}$$

Suppose otherwise that for some T_0, m_0 and $I_0 := (T_0, m_0)$ we have:

$$s-sound(T_0,T_0) \wedge (*T_0(I_0) \text{ halts}) \wedge (pr \circ T_0(I_0) \in \mathcal{O}) \wedge (pr \circ T_0(I_0) \text{ is } T_0\text{-stable}) \\ \wedge (R(\widetilde{T_0}(I_0)) \neq \infty).$$

Then by the above condition we get:

$$*T_0(I_0) \to (O,+), \text{ or } *T_0(I_0) \to (O,-), \tag{3.15}$$

for some $O = (X,\Sigma) \in \mathcal{O}$, which is (T_0,T_0)-stable and with property $sC(T_0,T_0)$, by $s-sound(T_0,T_0)$. We can of course guarantee that there is some m_0' with $T_0(T_0,m_0') = (O,+)$, but we arranged the details so that this is not necessary.

Since R is defined on all of $\mathcal{O}'$ we get:

$$R(\widetilde{T_0}(I_0)) = R(O,m_0) = X(\Sigma, m_0) = x \in \mathbb{Z}, \text{ for some } x,$$

by Property G of R and by $R(\widetilde{T_0}(I_0)) \neq \infty$. Then we have:

$$x = X(\Sigma, m_0) = D_1 \circ R \circ \widetilde{T_0}(I_0) = D_1(x) = x+1,$$

by (X,Σ) having property $sC(T,T)$, and by (3.15). So we get a contradiction and (3.14) follows. Our conclusion readily follows. □

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

Let S be a human subject in a controlled environment as before. For the moment S is not in any way idealized. We may then suppose, as in Section 2.2, that S determines an element of $\mathcal{M}^{\pm}$:

$$S : \mathcal{I} \to Strings \times \{\pm\}. \tag{4.1}$$

Once again S is only a partial function. As S now denotes two things: the human subject and the corresponding machine, we will say ***physical*** S when we want to clarify that we are talking of the actual human. In what follows when we say "stably assert", we mean that our physical S is will not change his mind. We may also just say ***perceive*** instead of stably assert, these words being technically synonymous in the usage here. On the other hand "assert" by itself is used in the usual sense of mathematicians asserting their theorems, possibly unsoundly.

We suppose S satisfies the condition that for any fixed $T \in \mathcal{T}^{\pm}$ if $(O,+) \in \{S(T,m)\}_m$ then at some point the physical S asserts O to have property $sC(S,T)$, while being part of the above environment.

This condition may appear to be self referential, since the physical S claims a property on strings that S prints, that refers to the machine S itself. So to be safe let us now rephrase this condition in a non self-referential manner.

For each $T \in \mathcal{T}^{\pm}$, a ***coherent system of machines*** $\{M_n(T)\}_{n\in\mathbb{N}}$ is defined to be a partial sequence of partial functions

$$M_n(T) : \{0,\dots,n\} \to Strings \times \{\pm\},$$

satisfying the ***coherency*** condition that

$$M_m(T)|_{\{0,\dots,n\}} = M_n(T)$$

for $m > n$. Let

$$M_\infty(T) : \mathbb{N} \to Strings \times \{\pm\}$$

denote the limiting partial function.

For each (T,n), we suppose that our physical S prints $S_n(T)$, in such a way that $\{S_n(T)\}_n$ is a coherent system of machines. And for each (T,n) S asserts:

$$\text{if } (O,+) = S_n(T)(n),$$

then for any abstract extension of $S_n(T)$ to a partial function

$$S^{ext}(T) : \mathbb{N} \to Strings \times \{\pm\},$$

$$\Theta_{S^{ext},T} \implies O \text{ has property } sC(T,T).$$

This property for each (T,n) now only logically depends on $S_n(T)$, which is defined, so there are no longer self referentiality issues.

Remark 4.2. *There may now be a worry that $\{S_n(T)\}_{T,n}$ could be an empty set. (Although this is technically irrelevant for us.) This is not the case however: since for any T we may start by*

$$S_1(T)(1) = (D_1 \circ R \circ \widetilde{T}, T, +).$$

So given $\{S_n(T)\}_{T,n}$ as above, we now rename back

$$S(T,n) := S_\infty(T)(n).$$

We also assume that if the physical S asserts that O has property $sC(S,T)$, while part of the environment above, then $(O,+) \in \{S(T,n)\}_n$. And so if the physical S perceives that O has property $sC(S,T)$ then O is stably (S,T)-printed.

Definition 4.3. *As before, we say that the physical S is* **computable** *if the corresponding machine S above is computable.*

By definition, S perceives that every (S,T)-stable O has property $sC(S,T)$, then as a consequence of the above, S must perceive by implication:

$$\forall T \in \mathcal{T}^{\pm} : \Theta_{S,T} \implies s-sound(T)$$

for S the above machine. And we ask that S is aware of Theorem 3.7, so that as a consequence S perceives that

$$\forall T \in \mathcal{T}^{\pm} : \Theta_{S,T} \implies \mathcal{G} \text{ has property } sC(T,T),$$

Which is the same as:

$$\forall T \in \mathcal{T}^{\pm} : \mathcal{G} \text{ has property } sC(S,T),$$

by the definition of property $sC(S,T)$. Then by our assumptions this is the same as S having stable Penrose property, defined as follows.

Definition 4.4. *We say that $M \in \mathcal{M}^{\pm}$ has* **the stable Penrose property** *or just* **Penrose property** *for short, if $\mathcal{G}$ is (M,T)-stable for every $T \in \mathcal{T}^{\pm}$.*

We further examine this property below. The following theorem is an immediate consequence of Theorem 3.7.

Theorem 4.5. *Let $M \in \mathcal{M}^{\pm}$ have the Penrose property then*

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

In fact, for any $M' \in \mathcal{T}^{\pm}$:

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

In particular if our physical S is computable, they cannot be fundamentally sound, specifically meaning stably sound.

4.1. **Examining the Penrose property.** This property is forced on our S by our assumptions. But these assumptions are based on our notion of "perceive", which is limited. To see this, for simplicity let us go back to discussing S in terms of statements in arithmetic they assert to hold, and as before we call this function as

$$P : \mathbb{N} \to \mathcal{A} \times \{\pm\},$$

for $\mathcal{A}$ sentences of arithmetic. Suppose then that the life of S is terminated. Then if $P(n) = (O,+)$ is the last output of P, O by our definition is P-stable. So that by our definition S perceives O as true, which is perhaps not what one ideally expects. And this makes the implication of our Theorem 4.5 weaker, since in this case clearly what S perceives to be true may not be true, so that there is no surprise that S is not fundamentally sound.

To make further sense of this, we could idealize our S so that their brain is not subject to deterioration. But then we have to understand the implications of our theorem for idealized S, for non-idealized S. In principle this shouldn't be a problem, since we can imagine a process of idealization that does not magically introduce non-computability in the limit, where the was none before. But this discussion is outside our scope.

Or we may say more concretely that the idealized human is represented by "the evolving scientific community" H, as we have already mentioned in the introduction. The fact that it is "evolving" because its members change (death/birth) presents no problems. If each individual human is Turing computable, then so is this H itself understood as machine:

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

where the output $H(I)$ is determined by, for example, majority consensus. On the other hand, by the same discussion as before H must perceive:

$$\forall T \in \mathcal{T}^{\pm} : \Theta_{H,T} \implies s-sound(T),$$

or more concretely H has the Penrose property. Then we may apply Theorem 4.5 above to this H, which then formalizes Theorem 0.1.

Proof of Theorem 4.5. Suppose $\Theta_{M,M'}$ for some $M' \in \mathcal{T}^{\pm}$. Suppose in addition $s - sound(M, M')$. Then by Theorem 3.7 for all m s.t. $pr \circ M(M', m)$ is (M, M')-stable:

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

but this contradicts the Penrose property. □

4.2. **The unreasonable complexity of a computable S.** Let us now not assume any idealizations of S. We want to give the reader some intuition into behavior of the function S, which may also give intuition for the theorem above, beyond the formal consequence.

Let $T \in \mathcal{T}^{\pm}$ be given to the physical S. Suppose that S is able to determine $T(T, n_0)$ for some n_0, before printing any of his strings. For example S may do this by brute force, using his computer to do the calculation $*T(T, n_0)$. Recall from Section 2.2 that S refers to the entire controlled environment of S, containing his computer. T computes S if it computes the answers of this system. So there is nothing wrong with S using his computer in this way. Or S may deduce $T(T, n_0)$ by some mathematical argument.

In this case, $S(T, n)$ for all n can have absolutely arbitrary values so long as $S(T, n_0) \neq T(T, n_0)$, since then automatically $\neg\Theta_{S,T}$, and so any string has property $sC(S, T)$! In other words if the physical S is able to a priori determine some value $T(T, n_0)$, then S immediately disproves $\Theta_{S,T}$, and so can print anything he wants for the values $S(T, n)$. By "a priori" we mean more specifically that the physical S has determined $T(T, n_0)$ before $S(T, n_0)$ is printed.

So if S is computable then any T computing S must be so complex that the physical S cannot a priori soundly determine any of T's values! Indeed, even without Theorem 4.5 this seems unlikely. The same would apply to an idealized S but becomes more extreme, since S is no longer time constrained.

5. Formal system interpretation

Let us fix our interpretation of an idealized human as H from before, postulated to have the Penrose property. As before, we will say physical H when we want to emphasize that we are talking of the physical thing representing the partial function H. Theorem 4.5, allows us to conclude that if H is computable then H not stably sound. The one string $\mathcal{G}$ that H is guaranteed to stably print, which expresses unsoundness of H, while elementary is slightly esoteric. Can we see more clearly that H is unsound? Yes, but we need stronger assumptions, and some language of formal systems. This section can be safely omitted as it is only of secondary interest.

For simplicity we will base everything of standard set theory $\mathcal{ST}$ (Zermelo-Fraenkel axioms). Turing machines, and arithmetic are assumed to be naturally formalized in $\mathcal{ST}$. In what follows, for a statement L, $\mathcal{F} \vdash L$ means that L is provable in the formal system $\mathcal{F}$.

Let $\mathcal{A}$ denote the set of sentences of arithmetic, as formalized by $\mathcal{ST}$. Let

$$P : \mathbb{N} \to \mathcal{A} \times \{+, -\},$$

be the machine associated to the physical H, analogously to the previous discussion, and as in the preamble to Section 3.

Definition 5.1. *We will say that the physical H* **is captured by a formal system $\mathcal{F} \supset \mathcal{ST}$** *if the following are satisfied:*

(1) *For any $T \in \mathcal{T}^{\pm}$:*

$$(O \text{ is } (H, T)\text{-stable}) \iff \mathcal{F} \vdash (O \text{ has property } sC(H, T)).$$

(2)

$$(A \in \mathcal{A} \text{ is } P\text{-stable}) \iff \mathcal{F} \vdash A.$$

We will suppose from now on that the physical H accepts $\mathcal{ST}$ as sound, meaning more specifically that H perceives its theorems as true, where the meaning of this is as in previous sense. In this case a formal system $\mathcal{F}$ in the language of set theory, that captures H, always exists. We may simply define

$\mathcal{F}(H)$ capturing H, by adding to $\mathcal{ST}$ as axioms the P-stable statements, and adding as an axiom, for each O (H,T)-stable, the statement:

$$O \text{ has property } sC(H,T).$$

The resulting formal system $\mathcal{F}(H)$ is in fact minimal, in the sense that any other $\mathcal{F}$ which captures H contains $\mathcal{F}(H)$. [9] Let $Con(H)$ denote the statement:

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

This is of course equivalent to the condition that $\mathcal{F}(H)$ is consistent by the above discussion.

Theorem 5.2. *Let H be as above then:*

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

In other words if H is provably computable then they are not consistent.

Note that the statement K:

$$\mathcal{ST} \vdash (\exists H' \in \mathcal{T}^{\pm} : \Theta_{H,H'})$$

does not mean that the physical H can prove the statement M:

$$\exists H' \in \mathcal{T}^{\pm} : \Theta_{H,H'} \tag{5.3}$$

in the practical sense. It just means that after the term H in the statement $\Theta_{H,H'}$ has been suitably interpreted in set theory $\mathcal{ST}$, M is provable in $\mathcal{ST}$. But a set theoretic, in other words mathematical, interpretation of the term H may not be practically attainable. At the least this necessitates detailed knowledge of the physics and biology underlying our humans. And even if this interpretation was attainable, H may not be clever enough to find the proof of M, again in the practical sense. Also note that $\neg Con(H)$ expresses *fundamental* inconsistency of H, as we only take stable assertions of H above.

Example 3. Suppose that H is provably computable as in the hypothesis of the theorem above. Then $\mathcal{F}(H)$ captures H and $\neg Con(\mathcal{F}(H))$ by the theorem, and so $\mathcal{F}(H)$ proves $0 = 1$. Since $\mathcal{F}(H)$ captures H the physical H must *stably* assert $0 = 1$.

Proof of Theorem 5.2. Let $\mathcal{F}$ capture H as above. By the proof of the second part of Theorem 4.5:

$$\mathcal{ST} \vdash ((\exists H' \in \mathcal{T}^{\pm} : \Theta_{H,H'}) \implies L),$$

where L is:

$$\exists m \in \mathbb{N}, \exists H' \in \mathcal{T}^{\pm} : (pr \circ H(H', m) \text{ is defined and is } (H,H')\text{-stable}) \wedge \\ \neg(pr \circ H(H', m) \text{ has property } sC(H,H')).$$

So

$$(\mathcal{ST} \vdash M) \implies (\mathcal{ST} \vdash L) \implies (\mathcal{F} \vdash L),$$

where M is as in (5.3).

On the other hand, by the assumption that H is captured by $\mathcal{F}$:

$$\mathcal{F} \vdash (\forall m \in \mathbb{N}, \forall H' \in \mathcal{T}^{\pm} : (pr \circ H(H', m) \text{ is defined and is } (H,H')\text{-stable}) \implies \\ (pr \circ H(H', m) \text{ has property } sC(H,H'))).$$

So $\mathcal{F} \vdash \neg L$. □

[9] $\mathcal{F}(H)$ may not however be effectively axiomatized, cf. Remark 3.2, but this is not relevant to our present discussion.

6. Relationship with the Penrose argument and its previous criticism

The most lucid criticism of the Penrose argument known to me appears in Koellner [15], [16]. Although our argument is very different it is worthwhile to consider it in light of that existing criticism. Brushing aside for the moment concerns such as soundness of human reasoning, the reason that the argument of Penrose is susceptible criticism is that it is only a meta-argument, meaning principally that it partly relies on interpretation of truth and knowledge. In other words, it is not written as a proof in some formal system, even implied one. And there may be logical subtleties to interpretation of truth that a non-logician such as myself or Penrose may simply miss.

So what of the main argument of this paper? Here the situation is rather different. With the possible exception of the inessential Section 5, where we touch on some logic, our theorems are in fact theorems of set theory, just classical Zermelo-Fraenkel set theory. For example our Gödel string $\mathcal{G}$ for a stably C-sound M is proved in set theory to have property sC. In contrast, in [20] Penrose asserts the truth of a certain analogously used Gödel statement by a meta-proof. And this is exactly what is contested by Koellner.

This is not to say that there are no issues of interpretation in this paper. One must interpret our definition of stable soundness as it applies to actual human beings. We of course have already partly addressed this. For the moment my position is that the idealized human being of this paper is represented by the world scientific community H (considered as evolving). And our theorem can be interpreted to directly apply to this H.

It is always the case that we must interpret mathematical theorems when applied to the real world. What one looks for is whether there is any meaningful physical obstruction to carrying out the necessary idealization in principle. In our specific case I see no such obstruction. Of course if the universe and humanity must eventually go extinct then our idealized humans cannot even in principle exist. But to me this is not a meaningful obstruction. The potential mortality of the universe is very unlikely to have any causal relation with computability of intelligence. So we can imagine an eternal universe, an immortal H and run the argument, then translate to our universe.

7. Concluding remark

While it can be argued that humans are not sound, it would be very difficult to argue that we are not stably sound. Scientists operate on the unshakeable faith that scientific progress converges on truth. And our interpretation above of this convergence as stable soundness is very simple and natural. Thus our results put a very serious obstruction to computability of intelligence.

In addition, at least under the stronger hypothesis of Theorem 5.2, stable unsoundness is testable/observable, at least in principle. For if H is provably computable as in Example 3 then H must eventually *stably* assert $0 = 1$. If we knew in addition that a given H' computes H then, as H' is a Turing machine, we can simulate it on a powerful computer and see if such non-sense statements really do appear. Given our basic understanding of humanity, such a possibility seems too ridiculous.

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

References

[1] A.M. Turing, *On computable numbers, with an application to the entscheidungsproblem* , Proceedings of the London mathematical society, s2-42 (1937).
[2] ———, *Computing machines and intelligence*, Mind, 49 (1950), pp. 433–460.
[3] L. Blum, M. Shub, and S. Smale, *On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines.*, Bull. Am. Math. Soc., New Ser., 21 (1989), pp. 1–46.
[4] D. J. Chalmers, *Minds machines and mathematics*, Psyche, symposium, (1995).
[5] B. J. Copeland and O. Shagrir, *Turing versus Gödel on computability and the mind.*, (2013), pp. 1–33.
[6] H. B. Curry, *Functionality in combinatory logic.*, Proc. Natl. Acad. Sci. USA, 20 (1934), pp. 584–590.
[7] D. Deutsch, *Quantum theory, the Church-Turing principle and the universal quantum computer.*, Proc. R. Soc. Lond., Ser. A, 400 (1985), pp. 97–117.

[8] S. Feferman, *Are There Absolutely Unsolvable Problems? Gödel's Dichotomy*, Philosophia Mathematica, 14 (2006), pp. 134–152.
[9] C. Fields, D. Hoffman, C. Prakash, and M. Singh, *Conscious agent networks: Formal analysis and application to cognition*, Cognitive Systems Research, 47 (2017).
[10] P. Grindrod, *On human consciousness: A mathematical perspective*, Network Neuroscience, 2 (2018), pp. 23–40.
[11] S. Hameroff and R. Penrose, *Consciousness in the universe: A review of the 'orch or' theory*, Physics of Life Reviews, 11 (2014), pp. 39 – 78.
[12] J.R. Lucas, *Minds machines and Gödel*, Philosophy, 36 (1961).
[13] K. Gödel, *Collected Works III (ed. S. Feferman)*, New York: Oxford University Press, 1995.
[14] A. Kent, *Quanta and qualia*, Foundations of Physics, 48 (2018), pp. 1021–1037.
[15] P. Koellner, *On the Question of Whether the Mind Can Be Mechanized, I: From Gödel to Penrose*, Journal of Philosophy, 115 (2018), pp. 337–360.
[16] ———, *On the question of whether the mind can be mechanized, ii: Penrose's new argument*, Journal of Philosophy, 115 (2018), pp. 453–484.
[17] K. Kremnizer and A. Ranchin, *Integrated information-induced quantum collapse*, Foundations of Physics, 45 (2015), pp. 889–899.
[18] R. Penrose, *Emperor's new mind*, 1989.
[19] ———, *Shadows of the mind*, 1994.
[20] ———, *Beyond the shadow of a doubt*, Psyche, (1996).

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