Direct link to author's version

STABLY SOUND TURING MACHINES AND INTELLIGENCE

YASHA SAVELYEV

Abstract. We first develop a mathematical notion of stable soundness intended to reflect the soundness property of (weakly idealized) human beings. Then we formally extend a disjunction of Gödel to show that given an abstract query machine M the following cannot hold simultaneously: M is stably sound, M is computable, M can stably decide the truth of any arithmetic statement. In practice this M is meant to represent a human being so that the above gives an obstruction to computability of intelligence. To this end, we develop in a specific setting an analogue of a Gödel statement for stably sound Turing machines.

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 make sense for humans. The human brain deteriorates and eventually fails, so that either we idealize the human brain to never deteriorate in particular die, or B now refers not to an individual human but to the evolving scientific community. We call such a human ***weakly idealized***.

[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.

Around the same time as Turing, Gödel argued for a no answer to Question 1, see [12, 310], relating the question to existence of absolutely unsolvable Diophantine problems, see also Feferman [7], and Koellner [14], [15] for a discussion. Essentially, Gödel argues for a disjunction:

$$\neg((S \text{ is computable}) \wedge (S \text{ is sound}) \wedge A),$$

where S refers to a certain idealized subject, and where A says that S can decide any Diophantine problem. This is in essence correct, that is can be formalized, see [15].

However, what is meaning of idealized above? If idealized just means stabilized in the sense of this paper then there is a Turing machine T whose stabilization T^s provably solves the halting problem, cf. Example 3.3, and so T^s is no longer a Turing machine. In that case, the above disjunction becomes meaningless because we introduced non-computability by passing to the idealization. So in this context one must be extremely detailed with what "idealized" means physically. The process of the physical idealization must be such that non-computability is not introduced in the ideal limit. In the case of weak idealization mentioned about it should certainly be possible, especially under the assumption of computability: the corresponding Turing machine / computer would be such a weak idealization! But this not what is needed by Gödel, he needs an idealization that is plausibly sound otherwise the disjunction would again be meaningless. In this case, it is absolutely not clear if and how this can work since we have no idea what is happening in the human brain.

Alternatively, one can try to enrich the argument of Gödel so that it explicitly allows for just fundamental soundness, understood here as stable soundness. But then we may worry: if stable soundness is such a loose concept that a Turing machine machine can stably soundly decide the halting problem, maybe Turing machines can stably soundly decide anything? No, we show here that there is a certain decision problem $\mathcal{P}$ that no Turing machine can stably soundly decide. And this is one ingredient for the following theorem.

After Gödel, Lucas [11] and later again and more robustly Penrose [17] argued for a no answer based only on soundness, and by further elaborating the obstruction from the Gödel incompleteness theorem. 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 [10] 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: [13], [16], [8], [9].

A number of authors, particularly Koellner [14], [15], argue that there are likely unresolvable meta-logical issues with the Penrose argument, even allowing for soundness. See also Penrose [17], and Chalmers [4] for discussions of some issues. After sincerely attempting to resolve these issues from a more elementary perspective (in the language of Turing machines) I concede that Koellner is right. The issue, as I see it, is loosely speaking the following. The kind of argument that Penrose proposes is a meta-algorithm P that takes as input specification of a Turing machine or a formal system, and has as output a natural number (or a string, sentence). Moreover, each step of this meta-algorithm is constructive (computably constructive). But the goal of the meta-algorithm P is to prove P is not computable as a function! So even on this surface level this appears impossible, unless P does something non-constructive, but then we must prove that our human can carry out this non-constructive step, and we are back to where we started.

What we argue here is that there is much more compelling version of the original Gödel disjunction, which only needs stable soundness. The following is a slightly informal version of our main Theorem 4.7.

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 stably unsound, or for any (could be weakly idealized) S there exists a certain true arithmetic statement, let's call it $\mathcal{H}(S)$*

[3], *that S will never stably soundly determine to be true. This theorem is indeed a mathematical fact* [4], *given our formalizations.*

By *absolutely* we mean in any sufficiently accurate 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 with Deutch [6], 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.

The thrust of the paper is to formally define stable soundness, and construct a new type of Gödel statements which works under this weaker hypothesis. Although our notion of stable soundness is general the Gödel statement is only constructed in a limited setting. Indeed it would be interesting to understand if this can be extended.

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 extended to stable diagonalization machines that print property C [5] strings only stably. This $\mathcal{G}$ is then used for the proof of the above theorem. Strictly speaking we can prove the theorem more directly but most of the setup needed for the construction of $\mathcal{G}$ would still be necessary, and using $\mathcal{G}$ makes the argument more conceptual. In addition it may be of independent interest.

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.*

[3] $\mathcal{H}(S)$ is a statement in the language of Turing machines and so is number theoretic, however it is not a Diophantine problem. Of course it cannot be, by Example 3.3.

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

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

- *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, 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 $T^1, \ldots, T^n$ 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:

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

with initial state $(q_0^1, q_0^2, 0)$ and final states:

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

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

$$(\sigma_i^1 \sigma_i^2), (\sigma_c^1 \sigma_c^2), (\sigma_o^1 \sigma_o^2)$$

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 unchanged, while leaving unchanged the symbols to which H_c^2, H_o^2 point.
(2) Move each head component H_i^1, H_c^1, H_o^1 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, $\sigma_i^2, \sigma_c^2, \sigma_o^2$ 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 unchanged, while leaving unchanged the symbols to which H_c^1, H_o^1 point.
(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, 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

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

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** [6] *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, see for instance [7]. 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:

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

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 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.*

[6]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.

Proof. Let W_n be some Turing machine $W_n : \{\epsilon\} \to \{\infty\}$, for $\epsilon \in \mathit{Strings}$ 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 $Z \in \mathit{Strings}$, 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 $Z = (X, \Sigma, m) \in \mathcal{O}' \subset \mathit{Strings}$ 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 $D \subset \mathcal{T} \times \mathit{Strings}$ 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 $D \supset \mathcal{O}'$. $\square$

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

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

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:

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

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

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 $O \in \mathit{Strings}$ 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* [5], *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 2.13. A trivially C-sound machine M is one for which

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

for every $(T, m) \in \mathcal{I}$. As $(D_1 \circ R \circ \widetilde{T}, T)$ automatically has property $C(M, T)$ for each $T \in \mathcal{T}_0$. In general, for any $M \in \mathcal{M}_0$, $T \in \mathcal{T}_0$ the list of all strings O with property $C(M, T)$ is always infinite, as by this example there is at least one such string $(D_1 \circ R \circ \widetilde{T}, T)$, which can then be modified to produce infinitely many such strings.

Theorem 2.14. *Given any $M \in \mathcal{M}_0$, if $sound(M, M') \land \Theta_{M,M'}$ 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 $\Theta_{M,M_0'} \land sound(M, M_0')$ 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:

$$\forall m, \forall T \in \mathcal{T}_0 : \Big(sound(T, T) \land (T(T, m) \in \mathcal{O}) \implies R(\widetilde{T}(T, m)) = \infty \Big). \tag{2.15}$$

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

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

So we have:

$$*\, T_0(I_0) \to (X, \Sigma) \in \mathcal{O}, \tag{2.16}$$

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 (2.16). So we get a contradiction and (2.15) follows. Our conclusion readily follows.

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

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

Also

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

therefore

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

Which is the same as:

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

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

□

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 is never deleted 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. So we have a machine:

$$P : \mathbb{N} \to \mathit{Strings} \times \{\pm\}.$$

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.*

Definition 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. We call this the* **stabilization** *of P. The range of P^s is called the* **stable output** *of P.*

In general P^s may not be computable even if P is computable. Explicit examples of this sort can be constructed by hand.

Example 3.3. We can construct a Turing machine

$$A : \mathbb{N} \to \mathit{Strings} \times \{\pm\},$$

whose stabilization A^s enumerates every Diophantine equation with no integer solution, or every Turing machine which does not halt. These sets are well known to be not computably enumerable, [1]. To do this we may proceed via a zig-zag algorithm.

In the case of Diophantine equations, here is a (inefficient) example. Let Z enumerate every polynomial with integer coefficients, and let N enumerate the integers.

- Initialize an ordered list L by $L = \emptyset$, which we understand as a list of instructions.
- Start. For each $p \in \{Z(0), \ldots, Z(n)\}$ check if $\{N(0), \ldots, N(n)\}$ are solutions of p. Whenever no add $(p,+)$ to L, whenever yes add $(p,-)$.

- Set $n := n + 1$ go to Start and continue.

This will define a partial function $A : \mathbb{N} \to \mathit{Strings}$ whose value $A(m)$ is the m'th, not necessarily final, instruction in the list L^m which is L after the m'th step of the algorithm. It's stabilization A^s enumerates polynomials which have no integer solutions.

We now translate the above 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

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

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 3.4. *For $M \in \mathcal{M}^{\pm}$ and for $(T, m) \in \mathcal{I}$, we say that an abstract $O \in \mathit{Strings}$ 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

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

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

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

$$\widetilde{M}(T, m) = (pr \circ M(T, m), m), \tag{3.5}$$

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

Definition 3.6. *Given a partial function $M : \mathit{Strings} \to \mathit{Strings} \times \{\pm\}$ and a Turing machine $T : \mathit{Strings} \to \mathit{Strings} \times \{\pm\}$, we say that T* **stably computes** *M if $M^s = T^s$ for T^s the stabilization of the partial function $fog(T)$.*

We write $\Theta^s_{M,T}$ for the statement T stably computes M. In what follows $\mathcal{O} \subset \mathcal{T}_{\mathbb{Z}} \times \mathit{Strings}$ is as before.

Definition 3.7. *For $M \in \mathcal{M}^{\pm}$, $M' \in \mathcal{T}^{\pm}$, an abstract string $O \in \mathit{Strings}$ is said to have property $sC = sC(M, M')$ if:*

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

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

for $\widetilde{M}'$ determined by M' as in (3.5).

Definition 3.8. *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 3.9. 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 3.10. *For all $M \in \mathcal{M}^{\pm}$:*

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

where

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

On the other hand,

$$\forall T \in \mathcal{T}^{\pm} : s - sound(T, T) \implies \mathcal{G} \text{ has property } sC(T, T). \tag{3.11}$$

Proof. This is mostly analogous to the proof of Theorem 2.14. Suppose not, let M be fixed and let M_0' be such that $s-sound(M, M_0') \wedge \Theta^s_{M,M_0'}$ and such that for some m_0:

$$M(M_0', m_0) = (\mathcal{G}, +) \text{ so that } \nexists n > m_0 : M(M_0', n) = (\mathcal{G}, -).$$

In particular $\mathcal{G}$ has property $sC(M, M_0')$ by $s - sound(M, M_0')$.

By $\Theta^s_{M,M_0'}$ there exists $m_0' > 0$ such that $*M_0'(M_0', m_0') \to (\mathcal{G}, +)$. If we set $I_0 := (M_0', m_0')$, then by $\mathcal{G}$ having property $sC(M, M_0')$, by $\mathcal{G} \in \mathcal{O}$ and by $\mathcal{G}$ being M_0'-stable as $\mathcal{G}$ is (M, M_0')-stable:

$$D_1(\infty, m_0) = D_1 \circ R \circ \widetilde{M_0'}(I_0). \tag{3.12}$$

On the other hand:

$$D_1 \circ R \circ \widetilde{M_0'}(I_0) = D_1 \circ R(D_1, \infty, m_0) \quad \text{by } M_0'(I_0) = (\mathcal{G}, +), \tag{3.13}$$

$$D_1 \circ R(D_1, \infty, m_0) = 2 \quad \text{by property } G \text{ of } R \text{ and by (2.2)}, \tag{3.14}$$

$$D_1(\infty, m_0) = 1, \tag{3.15}$$

$$1 = 2, \text{by (3.12) and by (3.14)}. \tag{3.16}$$

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:

$$s - sound(T, T) \wedge (pr \circ T(I) \in \mathcal{O}) \wedge (pr \circ T(I) \text{ is } T\text{-stable}) \implies R(\widetilde{T}(I)) = \infty. \tag{3.17}$$

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:

$$* T_0(I_0) \to (O, +), \text{ or } * T_0(I_0) \to (O, -), \tag{3.18}$$

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 (3.18). So we get a contradiction and (3.17) follows. Our conclusion readily follows. □

4. Stably undecidable problems and application

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 $\widetilde{S}(I)$. We are not actually assuming that $\widetilde{S}(I)$ is defined on every I.

So S is meant to determine an element of $\mathcal{M}^{\pm}$:

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

Remark 4.2. *The above is partially a simplification, because for a real world S it may be that each $\widetilde{S}(T, m)$ must be understood as a probability distribution on $Strings \times \{\pm\}$. In other words the value $\widetilde{S}(T, m)$ may only be 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. Moreover, we are only interested in stable output of $\widetilde{S}$ which, as we shall see from construction, should not be affected by any dice rolls.*

As S and $\widetilde{S}$ now denote 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 will not change his mind, formalized analogously to our definition of stably sound machines. 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 emphasize that although we talk of our S as a physical subject doing things like perceiving or asserting, we are just talking of various machines associated to this subject, analogously to $\widetilde{S}$ above, and of the mathematical properties of these machines. It is cumbersome to always make this explicit but implicitly we are always talking of set theoretic objects, in this section and elsewhere.

Let

$$D_S : \mathcal{T}^{\pm} \times \mathbb{N} \to (\{\hbar\} \bigsqcup \mathcal{U}) \times \{\pm\},$$

denote the machine with the properties below, for $\mathcal{U}$ an abstract set identified with $Strings$. We write $\mathcal{U}$ because otherwise the encoding

$$(\{\hbar\} \bigsqcup Strings) \times \{\pm\} \to Strings$$

may create confusion. But further on general elements of $\mathcal{U}$ may be implicitly identified with general elements of $Strings$. We ask that the following holds.

- For each T, n $D_S(T, n) = (O, +)$, with $O \in \mathcal{U}$, if S asserts at moment n that O is (T, T)-stable.
- For each T, n $D_S(T, n) = (\hbar, +)$ if S asserts at the moment n that no O is (T, T)-stable.
- For each T, n $D_S(T, n)$ is undefined if S at the moment n does not assert anything new.
- For each T, n $D_S(T, n) = (\hbar, -)$ if S no longer asserts at the moment n that no O is (T, T)-stable.

Let $\mathcal{D}$ denote the set of abstract machines of the form:

$$D : \mathcal{T}^{\pm} \times \mathbb{N} \to (\{\hbar\} \bigsqcup \mathcal{U}) \times \{\pm\}.$$

Definition 4.3. *We say that D is* **stably sound** *if for each T:*

$$\begin{aligned} \hbar \textit{ is } (D, T)\textit{-stable} &\implies \textit{no } O \textit{ is } (T, T)\textit{-stable} \\ O \in \mathcal{U} \textit{ is } (D, T)\textit{-stable} &\implies O \textit{ is } (T, T)\textit{-stable.} \end{aligned}$$

We say that D **stably decides** *$\mathcal{P}(T)$ if the partial function $D(T)$ has non-empty stable output. We say that D* **stably soundly decides** *$\mathcal{P}(T)$ if D is stably sound on T and stably decides $\mathcal{P}(T)$.*

For each $D \in \mathcal{D}$ there is an associated partial function

$$\widetilde{S}_D : \mathcal{T}^{\pm} \times \mathbb{N} \to Strings \times \{\pm\},$$

defined via the following meta-algorithm, which is a computational algorithm if D is a Turing machine.

- Let L be initialized as an empty set, which as before as an ordered list of instructions. Also initialize $n := 0$, and let $T \in \mathcal{T}^{\pm}$ be given.

- Start. If
$$D_S(T,n) = (O,+) \in \mathcal{U} \times \{\pm\} \text{ and } O \neq \mathcal{G}$$
then add $(O,-)$ and $(\mathcal{G},+)$ to L. If $O = \mathcal{G}$ then add $(O,-)$. If
$$D_S(T,n) = (\hbar,+)$$
then then add $(\mathcal{G},+)$ to L. If
$$D_S(T,n) = (\hbar,-)$$
then add $(\mathcal{G},-)$ to L. Finally, if $D_S(T,n)$ is undefined then do nothing.
- Set $n := n+1$ go to Start and continue.

The above determines a partial function $\widetilde{S}_D$ whose value $\widetilde{S}_D(T,m)$ is the m'th (not necessarily final) element of the list L^m which is L after the m'th iteration of the meta-algorithm. Unless L^m does not have at least m elements in which case we set $\widetilde{S}(T,m)$ to be undefined.

To summarize informally in terms of our physical S: if S perceives that some $O \neq \mathcal{G}$ is (T,T)-stable then $\widetilde{S} = \widetilde{S}_{D_S}$ satisfies that O is not $(\widetilde{S},T)$-stable, and so that $\mathcal{G}$ is $(\widetilde{S},T)$-stable. If S perceives that $\mathcal{G}$ (T,T)-stable then $\widetilde{S}$ satisfies that $\mathcal{G}$ is not $(\widetilde{S},T)$-stable. If S perceives that no O is (T,T)-stable then again $\mathcal{G}$ is $(\widetilde{S},T)$-stable. Finally if D_S does not stably decide $\mathcal{P}(T)$ then no O is $(\widetilde{S},T)$-stable.

The following is immediate by construction:

Proposition 4.4. *For $\widetilde{S}_D$ as above and for any $T \in \mathcal{T}^{\pm}$*
$$\neg\Theta^s_{\widetilde{S}_D,T} \vee \neg(D \textit{ stably soundly decides } \mathcal{P}(T)).$$

As a consequence we have:

Theorem 4.5. *There is no computable $D \in \mathcal{D}$ that stably soundly decides $\mathcal{P}$.*

Proof. Suppose otherwise that there is such a D, then by the above proposition we obtain:
$$\forall T \in \mathcal{T}^{\pm} : \neg\Theta^s_{\widetilde{S}_D,T},$$
but this is absurd since by construction $\widetilde{S}_D$ is computable if D is. $\square$

Definition 4.6. *We say that $\mathcal{A}(S)$ holds if for any $T \in \mathcal{T}^{\pm}$ whenever it is true that no O is (T,T) stable $\hbar$ is (D_S,T')-stable for some T' satisfying: $T \simeq_s T'$. Therefore, $\neg\mathcal{A}(S)$ in particular means that there exists a $T \in \mathcal{T}^{\pm}$ so that $\mathcal{H}(T)$:*
$$\textit{no } O \textit{ is } (T,T)\textit{-stable}$$
is true but S will never perceive it to be true.

$\mathcal{A}(S)$ is of course implied by S being able to perceive that a given Turing machine
$$f : \mathbb{N} \to \mathbb{N} \times \{\pm\}$$
has no f-stable output if f does not have f-stable output. However, the condition of the definition above is likely weaker since for T as above $T(T,\cdot)$ stably computes the partial function
$$u : \mathbb{N} \to \textit{Strings} \times \{\pm\},$$
which is nowhere defined. One can then hope, since u is so simple, that T can be put into a normal form T' with $T \simeq_s T'$ and with $T'(T',\cdot) = u$ and so that S can perceive that $T'(T',\cdot) = u$.

The following formalizes Theorem 0.1.

Theorem 4.7. *For S and D_S as above*
$$\neg((D_S \textit{ is stably sound}) \wedge (D_S \textit{ is computable}) \wedge \mathcal{A}(S)).$$

Proof. If D_S is stably sound then in particular $s - sound(\widetilde{S})$ by construction, for $\widetilde{S} = \widetilde{S}_{D_S}$. If D_S is computable then some T computes $\widetilde{S}$ by construction. In this case no O is (T,T)-stable since otherwise $\widetilde{S}$ has non-empty stable output, and by construction of $\widetilde{S}$ this can happen only if $\mathcal{G}$ is $\widetilde{S}$-stable in which case $\neg s - sound(\widetilde{S})$ by Theorem 3.10. So if $\mathcal{A}(S)$ then for some T' stably computing $\widetilde{S}$ $\hbar$ is $D_S(T')$-stable and so by construction of $\widetilde{S}$ $\mathcal{G}$ is $(\widetilde{S}, T')$-stable. But then $\mathcal{G}$ is (T,T)-stable which is a contradiction. □

5. Relationship with the Gödel and Penrose argument

The most lucid criticism of Gödel and Penrose arguments known to me appears in Koellner [14], [15]. Our argument is a certain significant extension of Gödel's argument, although we are also inspired by the ideas of Penrose. An important point is that our argument is entirely based on set theory, while Gödel's argument has meta-logical elements that require interpretation. Although, as Koellner explains [15], Gödel's argument can also be at least in some sense fully formalized.

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. At least under the previously explained assumption of weak idealization, stable soundness seems to be a very compelling hypothesis.

It is of course 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 weakly 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 and a weakly idealized human, run the argument then translate to our universe.

6. Concluding remark

While it can be argued that we 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. So the only thing to reasonably wonder is whether there could such a stably undecidable arithmetic statement of the form $\mathcal{H}(T)$ above. To this author this seems unlikely precisely because stable soundness is such a loose assumption, and mathematicians are so good at creating increasingly more powerful formal systems to reflect what they perceive to be true. However such a discussion is outside our scope.

Acknowledgements. Dennis Sullivan, Bernardo Ameneyro Rodriguez, David Chalmers, and in particular 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] H. B. Curry, *Functionality in combinatory logic.*, Proc. Natl. Acad. Sci. USA, 20 (1934), pp. 584–590.
[6] D. Deutsch, *Quantum theory, the Church-Turing principle and the universal quantum computer.*, Proc. R. Soc. Lond., Ser. A, 400 (1985), pp. 97–117.
[7] S. Feferman, *Are There Absolutely Unsolvable Problems? Gödel's Dichotomy*, Philosophia Mathematica, 14 (2006), pp. 134–152.
[8] C. Fields, D. Hoffman, C. Prakash, and M. Singh, *Conscious agent networks: Formal analysis and application to cognition*, Cognitive Systems Research, 47 (2017).
[9] P. Grindrod, *On human consciousness: A mathematical perspective*, Network Neuroscience, 2 (2018), pp. 23–40.
[10] 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.

[11] J.R. Lucas, *Minds machines and Gödel*, Philosophy, 36 (1961).
[12] K. Gödel, *Collected Works III (ed. S. Feferman)*, New York: Oxford University Press, 1995.
[13] A. Kent, *Quanta and qualia*, Foundations of Physics, 48 (2018), pp. 1021–1037.
[14] 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.
[15] ———, *On the question of whether the mind can be mechanized, ii: Penrose's new argument*, Journal of Philosophy, 115 (2018), pp. 453–484.
[16] K. Kremnizer and A. Ranchin, *Integrated information-induced quantum collapse*, Foundations of Physics, 45 (2015), pp. 889–899.
[17] R. Penrose, *Beyond the shadow of a doubt*, Psyche, (1996).

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