Direct link to author's version

INCOMPLETENESS FOR STABLY CONSISTENT TURING MACHINES DRAFT

YASHA SAVELYEV

Abstract. We first partly develop a mathematical notion of stable consistency intended to reflect the actual consistency property of human beings. Then we give a direct generalization of the first Gödel incompleteness theorem to stably (ω)-consistent Turing machines M. Our argument in particular proves the original (first) incompleteness theorem from first principles, using Turing machine language to construct our "Gödel sentence" directly, in particular we do not use the diagonal lemma. In practice such a stably consistent formal system could be meant to represent a weakly idealized human being so that the above gives an obstruction to computability of intelligence, and this gives a formal extension of a famous disjunction of Gödel.

1. Introduction

We begin by quickly introducing the notion of stable consistency. First, the term ***encoded map*** will mean a partial map $M : A \to B$, with A, B sets with an additional structure of an encoding in $\mathbb{N}$, (later on $\mathbb{N}$ is replaced by the set of strings in a finite alphabet). An encoding is just an injective map $e : A \to \mathbb{N}$ with some extra properties. This is described in more detail in Section 2.2. Working with encoded sets and maps as opposed to subsets of $\mathbb{N}$ will have the same advantages as working with abstract countably dimensional vector spaces as opposed to subspaces of $\mathbb{R}^\infty$.

Let $\mathcal{A}$ denote the set of (first order) sentences of arithmetic. And suppose we are given an encoded map

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

for $\{\pm\}$ a set with two elements $+, -$.

Definition 1.1.

- $\alpha \in \mathcal{A}$ **is** M**-stable** *if there is an* n_0 *with* $M(n_0) = (\alpha, +)$ *s.t. there is no* $m > n_0$ *with* $M(m) = (\alpha, -)$. *Let* $M^s : \mathbb{N} \to \mathcal{A}$ *enumerate in order of appearance the* M*-stable* α. *Abusing notation, we may also just write* M^s *for the set* $M^s(\mathbb{N})$, *where there is no risk of confusion.*
- M *is* **stably consistent** *if* M^s *is consistent.*
- M **decides arithmetic** *if*

$$\forall \alpha \in \mathcal{A} : (M^s \vdash \alpha) \vee (M^s \vdash \neg\alpha),$$

where $\vdash$ *means proves as usual.*
- M **decides arithmetic truth** *if*

$$\forall \alpha \in \mathcal{A} : (\alpha \textit{ is true}) \implies (M^s \vdash \alpha),$$

here by true we mean satisfied in the standard model of arithmetic.

So such an M could be given the following interpretation, this interpretation has no mathematical content a priori, it is simply how one may think of such an M informally. $M(n) = (\alpha, +)$ only if at the moment n M decides that α is true. Meanwhile

$$M(m) = (\alpha, -)$$

only if at the moment m, M no longer asserts that α is true. If α is M-stable we can also say M *stably asserts* α to be true. So that M is stably consistent if the set of sentences it stably decides to be true are consistent.

The following is the semantic version of our main theorem.

Theorem 1.2. *For M as above the following cannot hold simultaneously: M is stably consistent, M is computable and M decides arithmetic truth. Here 'computable' has the standard meaning of computability by a Turing machine, once one fixes encodings of the corresponding sets, with specifics given in Section 2.2.*

The above can be immediately deduced from Tarski undecidability of truth as the set $M^s(\mathbb{N})$ is definable in first order arithmetic whenever M is computable. However our proof is very elementary, starting with just the definition of Turing machine, in particular the diagonal lemma is not used. More crucially this proof readily extends to give the more interesting syntactic version of the theorem, extensively and directly generalizing the original Gödel incompleteness theorem. We present this generalization in the following.

Let RA denote Robinson arithmetic that is Peano arithmetic without induction. We say that $M : \mathbb{N} \to \mathcal{A} \times \{\pm\}$ is *stably ω-consistent* if it is stably consistent and if for any formula ϕ with one free variable, the following cannot happen simultaneously:

$$M^s \vdash \exists n : \phi(n),$$
$$\forall n : M^s \vdash \neg\phi(n).$$

Theorem 1.3. *For a stably ω-consistent, computable M as above, s.t. $M^s \supset RA$, there is an $\alpha \in \mathcal{A}$ which M^s can neither prove nor disprove, that is $\neg(M^s \vdash \alpha)$ and $\neg(M^s \vdash \neg\alpha)$. Moreover, the "Gödel sentence" α can be chosen to be a theorem of Zermelo-Fraenkel set theory about the standard structure $\mathbb{N}$ of natural numbers, or in other words $\mathbb{N}$ with its standard set theoretic interpretation.* [1]

This is a direct extension of the original Gödel incompleteness theorem, as we simply weaken the main assumption from consistency to stable consistency. We have preferred to avoid all meta-logic in formulation and the argument, so that the above is meant to simply be a theorem of Zermelo-Fraenkel set theory. This should make the above more accessible to general mathematical audience, but perhaps slightly less accessible to logicians.

After writing the present note, I discovered that there is a highly analogous theory by the name of 'experimental logics' in [10]. In relation to the latter paper there are two main differences. First to be we do not have the assumption corresponding to the convergence assumption in [10]. In our language 'convergence' says that if $(\alpha, +)$ is recurring in the output $M(\mathbb{N})$ then α is stable. This is a rather strong assumption for if we take M to represent the mathematical output of humanity (evolving in time), as we do in the discussion section further on, then there is no reason to suppose that any recurrent output of M is stable. The second difference is that we prove that the sentence α can be neither proved nor disproved by M^s, whereas in [10], it is only shown that the corresponding sentence is unprovable. In other words it is just the analogue of Theorem 1.2 above but with stronger assumptions. Finally our overall perspective and methods here are rather different, as we are interested in directly constructing the relevant (generalized) Gödel sentences from first principles, using the language of Turing machines.

If we choose to look at this theorem purely from the point of view of the set M^s of stable axioms, then the distinction with Gödel is that the set M^s may not be computably enumerable, see Example 3.3, whereas Gödel needs a computable sets of axioms. See [10] for an example of a Δ_2 definable (but non computably enumerable) extension of PA which can prove its own consistency statement. So in general there are obstructions to direct extensions of the incompleteness theorem even for definable sets of axioms, which M^s is. See also [13]. By imposing much stronger assumptions on consistency of the set of axioms, it is possible to give certain extensions [21], [15] but our theorem certainly does not fit into these extensions. Indeed, just from the viewpoint of the set M^s we just have the plain ω-consistency assumption on some non-computably enumerable set of axioms.

Our argument also readily reproves the original result of Gödel from first principles [2], with our Gödel sentence constructed directly by means of Turing machine language. One immediate question:

[1] We can likely convert this second half of the statement to say that α is true in the standard model $\mathbb{N}$ of arithmetic. This is not a priority for us, but should be a completely safe interpretation.

[2] Assuming existence of a certain encoding category $\mathcal{S}$, which is already essentially constructed by Gödel.

Question 1. Can we relax the condition of stable ω-consistency in the theorem above to just stable consistency, analogously to what is done by Rosser [20] for the original incompleteness theorem of Gödel?

The answer is not really obvious, because we use ω-consistency in a ostensibly stronger way then Gödel does, it is used to induce a certain additional technical property that we call 'speculative'.

1.1. **Motivational background - intelligence, Gödel's disjunction and Penrose.** 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 M is meant to be some system which contains a human subject. We do not care about what is happening inside M. 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 M produces given an input. Given this *very* limited interpretation, the question that we are interested in is this:

Question 2. 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. 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. However, we cannot honestly hope for consistency as even mathematicians are not on the surface consistent at all times. But we can certainly hope for some kind of fundamental consistency. In this note we will formally interpret fundamental consistency as stable consistency, which we already partly described above. This notion is meant to reflect the basic understanding of the way science progresses. Of course even stable consistency needs idealizations to make sense for individual humans. The human brain deteriorates and eventually fails, so that either we idealize the human brain to never deteriorate, or M now refers not to an individual human but to humanity, suitably interpreted. We call such a human ***weakly idealized***.

Remark 1.4. *In the case of humanity H, we may suppose that the output of the associated function*

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

is determined by majority consensus. This is not as restrictive as it sounds, for example if there is a computer verified proof of the Riemann hypothesis α in Zermelo-Fraenkel set theory, then irrespectively of the complexity of the proof, we can expect majority consensus for α, provided validity of set theory still has consensus. At least if we reasonably interpret H, which is beyond the scope here. In addition if stable consistency is explicitly the goal, (we can say that this is the experimental branch of human mathematical output) then H can safely consider increasingly more powerful axiomatic systems like Zermelo-Fraenkel + axiom of choice + continuum hypothesis, etc. By the same reasoning it is not at all unreasonable to suppose that H stably asserts all theorems of say Robinson arithmetic, it can just be interpreted as that H stably asserts all theorems generated by some particular Turing machine. This remark is primarily of interest in the context of Theorem 1.5 to follow.

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 [6], and Koellner [16], [17] for a discussion. Essentially, Gödel argues for a disjunction:

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

where S refers to a certain idealized subject, and where A says that S can decide any Diophantine problem. Gödel's argument can be formalized, see [17]. At the same time Gödel doubted that $\neg A$ is possible, again for an idealized S, as this puts absolute limits on what is humanly knowable even in arithmetic. Note that his own incompleteness theorem only puts relative limits on what is humanly knowable, within a fixed formal system.

However, what is the meaning of 'idealized' above? If idealized just means stabilized in the sense of this paper (Section 3 specifically) then there is a Turing machine T whose stabilization T^s soundly decides the halting problem, cf. Example 3.3, and so T^s is no longer computable. In that case, the above disjunction becomes meaningless because passing to the idealization may introduce non-computability where there was none before. So in this context one must be extremely detailed with what "idealized" means physically and mathematically. The process of the idealization must be such that non-computability is not introduced in the ideal limit. For weak idealization in terms of humanity mentioned above this is automatic, for the more direct idea of idealizing brain processes it should certainly also be in principle possible. For example, suppose we know that there is a mathematical model M for the biological human brain, in which the deterioration mentioned above is described by some explicit computable stochastic process, on top of the base cognitive processes. Then mathematically a weak idealization M^i of M would just correspond to the removal of this stochastic process. We may then meaningfully apply the above theorem to M^i, since M^i would be non-computable only if some cognitive processes of the brain (in the given mathematical model) were non-computable, since mathematically M^i would be composed from such processes.

But this is not what is needed by Gödel. He needs an idealization that is plausibly consistent otherwise the disjunction would again be meaningless, while a weak idealization of a human is only plausibly stably consistent. Since we do not have a good understanding of physical processes of the human brain involved in cognition, it is not at all clear that what Gödel asks is even possible.

So the natural solution to attempt is to enrich the argument of Gödel so that it explicitly allows for just stable consistency. But then we may worry: if stable consistency/soundness is such a loose concept that a stably sound Turing machine can decide the halting problem, maybe Turing machines can stably consistently decide anything? So we need new incompleteness theorems.

After Gödel, Lucas [11] and later again and more robustly Penrose [19] argued for a no answer based only on soundness and the Gödel incompleteness theorem, that is attempting to remove the necessity to decide A or $\neg A$. A number of authors, particularly Koellner [16], [17], argue that there are likely unresolvable meta-logical issues with the Penrose argument, even allowing for soundness. See also Penrose [19], and Chalmers [4] for discussions of some issues. 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 computably constructive. But the goal of the meta-algorithm P is to prove P is not computable as a function! So on this rough surface level this appears to be impossible.

Notwithstanding, what we argue here is that there is more compelling version of the original Gödel disjunction that only needs stable consistency. The following is a slightly informal, applied version of our Theorems 1.2, 1.3. In what follows, H refers to an encoded map as in the first part of introduction, associated to humanity as discussed. So this is our present concrete model for a weakly idealized human, we also theorized above other perhaps neater models for a weakly idealized human, and the theorem could also be stated in that context, but this is beyond our scope.

Theorem 1.5. *Suppose that our model for humanity $H : \mathbb{N} \to \mathcal{A} \times \{\pm\}$, stably asserts all theorems of Robinson arithmetic, and is stably ω-consistent. Then either there are cognitively meaningful, absolutely non Turing computable processes in the human brain, or there exists a statement of arithmetic $\mathcal{H}$ such that H will never stably decide $\mathcal{H}$ or $\neg\mathcal{H}$. Moreover $\mathcal{H}$ can be assumed to be true in the standard model of arithmetic.*

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]) [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 [5], 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.

Remark 1.6. *Turing suggested in [2] that abandoning hope of consistency is the obvious way to circumvent the implications of Gödel incompleteness theorem. In my opinion this position is untenable, humans (reasonably idealized) may not be consistent but it is inconceivable that humanity is not stably consistent, and in stably consistent case there is still an incompleteness theorem. So given stable consistency, the only way that the incompleteness theorems can be circumvented is to accept that there is an absolute limit on the power of human reason as in the theorem above.*

Remark 1.7. *It should also be noted that for Penrose, in particular, non-computability of intelligence would be evidence for new physics, and he has specific and* very *intriguing proposals with Hameroff* [9] *on how this can take place in the human brain. As we have already indicated, new physics is not a logical necessity for non-computability of brain processes, at least given the state of the art. However, it is very plausible that new physical-mathematical ideas may be necessary to resolve the deep mystery of human consciousness. Here is also a partial list of some partially related work on mathematical models of brain activity, consciousness and or quantum collapse models:* [14], [18], [7], [8].

2. Some preliminaries

This section can be just skimmed on a first reading. For more details we recommend the book of Soare [**?**]. Our approach here is however is slightly novel in that we do not explicitly Gödel encode anything, instead abstractly axiomatizing the expected properties of encodings. This allows us later on to give very concise, self-contained arguments for main results.

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 2.1. *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 *with at least two elements. 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).*

[3] We are now involving real numbers but there is a standard way of talking of computability in this case, in terms of computable real numbers. This is what means over $\mathbb{Z}$.

   (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 2.2. *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, $\delta^M(s)$ will denote the successor state of s, obtained by executing the instructions set of M on s, or in other words $\delta^M(s)$ 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 2.3. *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**. *For a given Turing computation $*M(\Sigma)$, we will write*

$$*M(\Sigma) \to x,$$

if $*M(\Sigma)$ halts and x is the corresponding 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. Denote by *Strings* the set of all finite strings of symbols in Γ, including the empty string ϵ. Then a Turing machine M determines a partial function that is defined on all $\Sigma \in Strings$ s.t. $*M(\Sigma)$ halts, by $\Sigma \mapsto M(\Sigma)$.

In practice, it will be convenient to 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

$$\Sigma \notin \mathcal{I} \subset Strings.$$

The set $\mathcal{I}$ is also called the set of T-***permissible*** input strings. We do not ask that for $\Sigma \in \mathcal{I}$ $*T(\Sigma)$ halts. If $*T(\Sigma)$ does halt then we will say that Σ is T-***acceptable***.

Definition 2.4. *We denote by $\mathcal{T}$ the set of all Turing machines with a distinguished final machine state q_{reject}.*

It will be convenient to forget q_{reject} and instead write

$$T : \mathcal{I} \to \mathcal{O},$$

where $\mathcal{I} \subset Strings$ is understood as the subset of all T-permissible strings, or just ***input set*** and $\mathcal{O}$ is the set output strings or ***output set***.

Definition 2.5. *Given a partial function*

$$f : \mathcal{I} \to \mathcal{O},$$

we say that a Turing machine $T \in \mathcal{T}$

$$T : \mathcal{I} \to \mathcal{O}$$

computes *f if $T = f$ as partial functions on $\mathcal{I}$.*

2.1. **Multi-input Turing machine.** There is a basic well known variant of a Turing machine which takes as input an element of $Strings^n$, for some fixed $n \in \mathbb{N}$. This is done by replacing the input 1-tape by an n-tape. We will not give details of this. Notationally such a Turing machine will be distinguished by a superscript so

$$T^n : Strings^n \to Strings,$$

denotes such a Turing machine with n inputs.

2.2. **Abstractly encoded sets.** The material of this section will be used in the main arguments, it will allow us to remove the need to work with explicit Gödel encodings, greatly simplifying subsequent details. However this will require a bit of abstraction. This is analogous in linear algebra to working with abstract $\mathbb{R}$-vector spaces as opposed to $\mathbb{R}^n$. While formally one of course gains nothing, any student of linear algebra appreciates the power gained by introducing the abstraction of vector spaces.

An ***encoding*** of a set A is an injective set map $e : A \to Strings$. For example we may encode $Strings^2$ as a subset of $Strings$ as follows. The encoding string of $\Sigma = (\Sigma_1, \Sigma_2) \in Strings^2$ will be of the type: "this string encodes an element $Strings^2$: its components are Σ_1 and Σ_2." In particular the sets of integers $\mathbb{N}, \mathbb{Z}$, which we may use, 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.

More formally, let $\mathcal{S}$ be a small arrow category whose objects are maps $e_A : A \to Strings$, for e_A an embedding called ***encoding map of*** A, determined by a set A. We may denote $e_A(A)$ by A_e. The morphisms in the category $\mathcal{S}$ are pairs of partial maps (m, m_e) so that the following diagram commutes:

$$\begin{array}{ccc} A & \xrightarrow{\quad m \quad} & B \\ \downarrow{\scriptstyle e_A} & & \downarrow{\scriptstyle e_B} \\ A_e \subset Strings & \xrightarrow{m_e} & B_e \subset Strings, \end{array}$$

and so that m_e is computable and A_e coincides with the set of permissible strings for the Turing machine computing m_e.

Notation 1. We may just write $A \in \mathcal{S}$ for an object, with e_A implicit.

We call such an A an ***abstractly encoded set*** so that $\mathcal{S}$ is a category of abstractly encoded sets. In addition we ask that $\mathcal{S}$ satisfies the following properties.

(1) For $A \in \mathcal{S}$ A_e is computable (recursive). Here, as is standard, a set $S \subset Strings$ is called *computable* if both S and its complement are computably enumerable, with S called *computably enumerable* if there is a Turing machine T so that $*T(\Sigma)$ halts iff $\Sigma \in S$.
(2) There is an abstractly encoded set $\mathcal{U} = Strings \in \mathcal{S}$, with $e_{\mathcal{U}} = id_{Strings}$. We can think of $\mathcal{U}$ as the set of typeless strings.
(3) For $A, B \in \mathcal{S}$,
$$(A_e \cap B_e \text{ is non-empty}) \implies (A = \mathcal{U}) \vee (B = \mathcal{U}).$$
In particular each $A \in \mathcal{S}$ is determined by A_e.
(4) If $A, B \in \mathcal{S}$ then $A \times B \in \mathcal{S}$ and the projection maps $pr^A : A \times B \to A$, $pr^B : A \times B \to B$ complete to morphisms of $\mathcal{S}$, similar to the above, so that we have a commutative diagram:
$$\begin{array}{ccc} A \times B & \xrightarrow{pr^A} & A \\ \downarrow & & \downarrow \\ (A \times B)_e & \xrightarrow{pr^A_e} & A_e, \end{array}$$
with pr^A_e computable, similarly for pr^B. In addition we ask for the following naturality property. (We don't strictly speaking use this last property, but it may be helpful for understanding $\mathcal{S}$.) Let f be the composition
$$A_e \times B_e \xrightarrow{(e_A^{-1}, e_B^{-1})} A \times B \xrightarrow{e_{A \times B}} (A \times B)_e,$$

then f is computable meaning that there is a 2-input Turing machine T^2, with input set $A_e \times B_e$ such that

$$T^2(\Sigma_1, \Sigma_2) = f(\Sigma_1, \Sigma_2)$$

for all (Σ_1, Σ_2) in $A_e \times B_e$.

The above axioms suffice for our purposes but there are a number of possible extensions. The specific such category $\mathcal{S}$ that we need will be clear from context later on. We only need to encode finitely many types of specific sets. For example $\mathcal{S}$ should contain an abstract encoding of $\mathbb{Z}, \mathbb{N}, \{\pm\}, \mathcal{T}$, with $\{\pm\}$ a set with two elements $+, -$. The encodings of $\mathbb{N}, \mathbb{Z}$ should be suitably natural so that for example the map

$$\mathbb{N} \to \mathbb{N}, \quad n \mapsto n+1$$

completes to a morphism in $\mathcal{S}$. For $\mathbb{Z}$ we also want the map

$$\mathbb{Z} \to \mathbb{Z} \quad n \mapsto -n$$

to complete to a morphism in $\mathcal{S}$.

Definition 2.6. *An* **abstract Turing machine** $T : A \to B$*, will just be another name for a morphism in the category* $\mathcal{S}$*. So that really this is a pair* (T, T_e)*, with* T_e *an implicit Turing machine computing* $e_B \circ T \circ e_A^{-1}$*. We may just say Turing machine in place of abstract Turing machine, since it is usually clear that what we mean.*

We define $\mathcal{T}_{\mathcal{S}}$ to be the set of abstract Turing machines relative to $\mathcal{S}$ as above. $\mathcal{T}_{\mathcal{S}}$ in general cannot be naturally encoded, but there is a natural embedding $i : \mathcal{T}_{\mathcal{S}} \to \mathcal{T}$, $i(T) = T_e$ and $\mathcal{T}$ is encoded, and this will be sufficient for us.

For writing purposes we condense the above as follows.

Definition 2.7. *An* **encoded map** *will be a synonym for a partial map* $M : A \to B$*, with* A, B *abstractly encoded sets.*

$\mathcal{M} = \mathcal{M}_{\mathcal{S}}$ will denote the set of encoded maps. Given an abstract Turing machine $T : A \to B$, we have an associated encoded map $fog(T) : A \to B$ defined by forgetting the additional structure T_e. However we may also just write T for this encoded map by abuse of notation. So we have a forgetful map

$$fog : \mathcal{T} \to \mathcal{M},$$

which forgets the extra structure of a Turing machine.

Definition 2.8. *We say that an abstract Turing machine* T **computes** $M \in \mathcal{M}$ *if* $fog(T) = M$*. We say that* M *is* **computable** *if some* T *computes* M*.*

2.3. **Notation.** $\mathbb{Z}$ always denotes the set of all integers and $\mathbb{N}$ non-negative integers. We will sometimes specify an (abstract) Turing machine simply by specifying a map

$$T : \mathcal{I} \to \mathcal{O},$$

with the full data of the underlying Turing machine being implicitly specified, in a way that should be clear from context. We will not notationally distinguish naturals $n \in \mathbb{N}$ from their corresponding numerals in the language of arithmetic, as this usually will not lead to any confusion.

3. On stable consistency and soundness

Definition 3.1. *Given an encoded map:*

$$M : \mathbb{N} \to B \times \{\pm\},$$

We say that $b \in B$ **is** M**-stable** *if there is an* n_0 *with* $M(n_0) = (b, +)$ *s.t. there is no* $m > n_0$ *with* $M(m) = (b, -)$*. In the case above, we may also say that* M **prints** b **stably**.

Definition 3.2. *Given a encoded map*

$$M : \mathbb{N} \to B \times \{\pm\},$$

we define

$$M^s : \mathbb{N} \to B$$

to be the machine enumerating, in order, all the M-stable b. We call this the **stabilization** *of M. The range of M^s is called the* **stable output** *of M.*

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

Example 3.3. We can construct an abstract Turing machine

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

whose stabilization A^s enumerates every Diophantine (integer coefficients) polynomial with no integer roots, where $\mathcal{P}$ denotes the set of all Diophantine polynomials, (also abstractly encoded). Similarly, we can construct a Turing machine D whose stabilization enumerates pairs (T, n) for $T : \mathbb{N} \to \mathbb{N}$ a Turing machine and $n \in \mathbb{N}$ such that $*T(n)$ does not halt. In other words D stably soundly decides the halting problem. To do this we may proceed via a zig-zag algorithm.

In the case of Diophantine polynomials, here is a (inefficient) example. Let Z computably enumerate every Diophantine polynomial, and let N computably enumerate the integers. In other words, in our language, $Z : \mathbb{N} \to \mathcal{P}$, $N : \mathbb{N} \to \mathbb{Z}$ are total bijective functions extending to morphisms of $\mathcal{S}$ (in particular abstract Turing machines). The encoding of $\mathcal{P}$ should be suitably natural so that in particular the map

$$E : \mathbb{Z} \times \mathcal{P} \to \mathbb{Z}, \quad (n, p) \mapsto p(n)$$

extends to a morphism in $\mathcal{S}$. In what follows, for each $n \in \mathbb{N}$, L_n has the type of an ordered finite list of elements of $\mathcal{P} \times \{\pm\}$, with order starting from 0.

- Initialize $L_0 := \emptyset$, $n := 0$.
- Start. For each $p \in \{Z(0), \ldots, Z(n)\}$ check if one of $\{N(0), \ldots, N(n)\}$ is a solution of p, if no add $(p, +)$ to L_n, if yes add $(p, -)$. Call the resulting list L_{n+1}. Explicitly,

  $$L_{n+1} := L_n \cup \bigcup_{m=0}^{n} (Z(m), d^n(Z(m))),$$

  where $d^n(p) = +$ if none of $\{N(0), \ldots, N(n)\}$ are roots of p, $d^n(p) = -$ otherwise,

  where $\cup$ is set union operation of subsets of $\mathcal{P} \times \{\pm\}$.
- Set $n := n + 1$ go to Start and continue.

Set $L := \cup_n L_n$, which is an ordered infinite list. Define $A : \mathbb{N} \to \mathcal{P} \times \{\pm\}$ by $A(m) := L(m)$, with the latter the m'th element of L. Since E is computable, it clearly follows that A is computable and its stabilization A^s enumerates Diophantine polynomials which have no integer roots.

3.1. Decision maps.

By a ***decision map*** we mean an encoded map of the form:

$$D : B \times \mathbb{N} \to \{\pm\}.$$

Definition 3.4. *For a D as above we say that $b \in B$ is D*-**decided** *if there is an N s.t. for all $n \geq N$ $D(b, n) = +$.*

Given a total encoded map $M : \mathbb{N} \to B \times \{\pm\}$, there is an associated total decision map:

$$D_M : B \times \mathbb{N} \to \{\pm\},$$

Define $b \in B$ to be n-stable if there a m, $0 \leq m \leq n$, with $M(m) = (b, +)$ s.t. there is no k satisfying $0 < k \leq n$, with $M(k) = (b, -)$. D_M is then defined by

$$D_M(b, n) = \begin{cases} + & \text{if } b \text{ is } n\text{-stable} \\ - & \text{otherwise.} \end{cases}$$

Lemma 3.5. *If M as above is computable then D_M is computable. Moreover, b is M-stable iff b is D_M-decided.*

Proof. If M is computable then the set

$$\mathcal{B} = \{(b, n) \in B \times \mathbb{N} \,|\, b \text{ is } n\text{-stable}\}$$

is obviously Turing decidable and so D_M is computable. The second part of the lemma is immediate. □

In particular by the example above there is a Turing machine

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

that stably soundly decides if a Diophantine polynomial has integer roots, meaning:

$$p \text{ is } D_A\text{-decided} \iff p \text{ has no integer roots.}$$

We can of course also construct such a D_A more directly. Likewise there is a Turing decision machine that stably soundly decides the halting problem, in this sense.

Definition 3.6. *Given an encoded map*

$$M : B \times \mathbb{N} \to \{\pm\}$$

and a Turing machine

$$T : B \times \mathbb{N} \to \{\pm\},$$

we say that T **stably computes** *M, or that $\Theta_{M,T}$ holds, if*

$$b \text{ is } M\text{-decided} \iff b \text{ is } T\text{-decided.}$$

Definition 3.7. *Given Turing machines*

$$T_1, T_2 : B \times \mathbb{N} \to \{\pm\},$$

we say that they are **stably equivalent** *and write $T_1 \simeq_s T_2$ if T_1, T_2 stably compute the same machine.*

3.2. **Preliminaries on arithmetic decision maps.** In what follows M denotes a total encoded map $M : \mathbb{N} \to \mathcal{A} \times \{\pm\}$.

Lemma 3.8. *Given M, there is an encoded map $CM : \mathbb{N} \to \mathcal{A} \times \{\pm\}$ so that $CM^s(\mathbb{N})$ is the deductive closure of the set $M^s(\mathbb{N})$, and so that if M is computable then so is CM.*

This might be rather evident but a formal argument requires some care.

Proof. Given a Turing machine $T : \mathbb{N} \to \mathcal{A}$, there is a Turing machine $CT : \mathbb{N} \to \mathcal{A} \times \mathcal{P}$, where $\mathcal{P}$ is the set of proofs in the formal system T, with the property that $pr_{\mathcal{A}} \circ CT$ enumerates the deductive closure of $T(\mathbb{N})$. And so that, for each n, $pr_{\mathcal{P}} \circ CT(n)$ is the proof of $pr_{\mathcal{A}} \circ CT(n)$ in the formal system $\{T(0), \ldots, T(n)\}$. Here, $pr_{\mathcal{A}} : \mathcal{A} \times \mathcal{P} \to \mathcal{A}$ and $pr_{\mathcal{P}} : \mathcal{A} \times \mathcal{P} \to \mathcal{P}$ are the natural projections. Existence of CT is elementary and well known.

Denote by $pr_{\mathcal{A}} : \mathcal{A} \times \{\pm\} \to \mathcal{A}$, and $pr_{\pm} : \mathcal{A} \times \{\pm\} \to \{\pm\}$ the pair of projections, set $M' := pr_{\mathcal{A}} \circ M$.

Definition 3.9. *Let $\alpha \in \mathcal{A}$, then α will be called n-***stable** *(with respect to M) if the following holds.*

- *There exists $m \leq n$ s.t. $CM'(m) = (\alpha, p)$ so p is a proof of α in $F = \{\alpha_1, \ldots, \alpha_k\}$ with $\alpha_1, \ldots, \alpha_k \in \operatorname{image} M'$.*
- *All α_i, $1 \leq i \leq k$ as above satisfy: $\exists d \in \{0, \ldots, n\} : M(d) = (\alpha_i, +)$, and there is no $m > d$ with $M(m) = (\alpha_i, -)$.*

For each $n \in \mathbb{N}$, L_n will have the type of an ordered finite list of elements in $\mathcal{A} \times \{\pm\}$.

(1) Initialize $L_0 = \emptyset$ and $n := 0$.
(2) Start. Set
$$U_n := \{pr_{\mathcal{A}}\sigma \,|\, \sigma \in L_n,\ pr_{\pm}\sigma = +,\ pr_{\mathcal{A}}\sigma \text{ is not } n\text{-stable}\}.$$
Set $L'_{n+1} := L_n \cup \bigcup_{\alpha \in U_n} \{(\alpha, -)\}$ and set $L_{n+1} := L'_{n+1} \cup \{(CM'(n), +)\}$, if $CM'(n)$ is n-stable, otherwise set $L_{n+1} := L'_{n+1}$.

(3) $n := n + 1$, go to Start.

The above recursion gives an ordered infinite list $L := \cup_n L_n$, for each n set $CM(n)$ to be the n-th element of the list. □

Definition 3.10. *We say that M is* **speculative** *if the following holds. Let ϕ be a formula in arithmetic with one free variable, such that for each n $\phi(n)$ is M^s-decidable. Let α be the sentence:*

$$\forall m : \phi(m),$$

then

$$\forall m : M^s \vdash \phi(m) \implies M^s \vdash \alpha.$$

Note that of course the left hand side is not the same as $M^s \vdash \alpha$.

We may informally interpret this condition as saying that M initially prints α as a hypothesis, and removes α from its list (that is α will not be in M^s) only if for some m, $M^s \vdash \neg\phi(m)$. Since each $\phi(m)$ is by assumption M^s-decidable, this is not in principle conflictory with stable consistency. Indeed in the example above we construct a stably sound Turing machine, with an analogue of this speculative property, deciding the halting problem. Moreover, we have the following.

Theorem 3.11. *Given any encoded map*

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

so that $M^s \supset RA$ there is a speculative encoded map

$$M_{spec} : \mathbb{N} \to \mathcal{A} \times \{\pm\}$$

with the properties: if M is computable so is M_{spec}, $M^s_{spec}(\mathbb{N}) \supset M^s(\mathbb{N})$, if M is stably ω-consistent then M_{spec} is stably ω-consistent.

Thus the speculative condition is not as forcing as it may sound.

Proof. Let M be as in the hypothesis. Let $\mathcal{F}$ denote the set of formulas ϕ in arithmetic with one free variable. We encode $\mathcal{F}$ so that the map $\mathcal{F} \to \mathcal{A}$, $\phi \mapsto \phi(n)$ is computable. With this understanding $\mathcal{F} \in \mathcal{S}$, where $\mathcal{S}$ is our category of encoded sets.

Lemma 3.12. *There is a Turing machine $F : \mathbb{N} \to \mathcal{F} \times \{\pm\}$ with the property:*

$$F^s(\mathbb{N}) = G := \{\phi \in \mathcal{F} \,|\, \forall n : M^s \vdash \phi(n)\}.$$

Proof. The construction is analogous to the construction in the Example 3.3 above. Let CM be as in Lemma 3.8, for brevity set

$$Q := CM.$$

Let

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

be the associated Turing machine as in Lemma 3.5. Fix any total bijective Turing machine

$$Z : \mathbb{N} \to \mathcal{F}.$$

For an $\alpha \in \mathcal{A}$ we will say that it is n-***decided*** if there is a k, $0 \leq k \leq n$, s.t. $D_Q(\alpha, m) = +$ for all m satisfying $k < m \leq n$. In what follows each L_n has the type of an ordered finite list of elements of $\mathcal{F} \times \{\pm\}$.

- Initialize $L_0 := \emptyset$, $n := 0$.
- Start. For each $\phi \in \{Z(0), \ldots, Z(n)\}$ if $\phi(m)$ is n-decided for all $0 \leq m \leq n$, then add $(\phi, +)$ to the list L_n, otherwise add $(\phi, -)$ to L_n. Explicitly, we set

$$L_{n+1} := L_n \cup \bigcup_{\phi \in \{Z(0), \ldots, Z(n)\}} (\phi, d^n(\phi)),$$

where $d^n(\phi) = +$ if ϕ is n-decided and $d^n(\phi) = -$ otherwise.

Here $\cup$ is set union operation of subsets of $\mathcal{F} \times \{\pm\}$.

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

Set $L := \cup_n L_n$, which is an ordered infinite list. Define
$$F : \mathbb{N} \to \mathcal{F} \times \{\pm\}$$
by
$$F(m) := L(m),$$
with the latter the m'th element of L. Clearly F is computable. □

Define:
$$M_{spec}(n) := \begin{cases} M(k) \text{ if } n = 2k+1 \\ F(k) \text{ if } n = 2k. \end{cases}$$
Then M_{spec} is speculative and computable. Set
$$S := M^s_{spec}(\mathbb{N}).$$
Then S is consistent unless for some $\phi \in G$
$$M^s \vdash \neg \forall n : \phi(n),$$
that is
$$M^s \vdash \exists n : \neg \phi(n).$$
Since M^s is ω-consistent this implies:
$$\exists n_0 \in \mathbb{N} : M^s \vdash \neg \phi(n_0),$$
but since $\phi \in G$ this means that M^s is inconsistent, a contradiction, so S is consistent.

Lemma 3.13. *If M is speculative, $M^s \supset RA$, and M is stably consistent then it is stably ω-consistent.*

Proof. Suppose
$$\forall n : M^s \vdash \neg \phi(n),$$
for ϕ as in the statement of ω-consistency. So by the speculative property:
$$M^s \vdash (\forall n : \neg \phi(n)).$$
Then it cannot be that
$$M^s \vdash \exists n : \phi(n),$$
as otherwise M^s is not consistent. Thus M must be stably ω-consistent. □

Our theorem then follows by S being consistent and by the lemma above. □

4. Incompleteness for stably sound Turing machines

We first warm up with the simpler case of stable soundness, as we will also use most of the concepts here for stable consistency.

Let $\mathcal{D}$ denote the set of total encoded maps of the form:
$$D : \mathcal{T} \times \mathbb{N} \to \{\pm\}.$$
And set
$$\mathcal{D}^t := \{T \in \mathcal{T}_\mathcal{S} | fog(T) \in \mathcal{D}\}.$$
In what follows, for $T \in \mathcal{T}$ when we write $T \in \mathcal{D}^t$ we mean that $T \in \operatorname{image} i|_{\mathcal{D}^t}$, for $i : \mathcal{T}_\mathcal{S} \to \mathcal{T}$ the embedding discussed in Section 2.2. Note that $\operatorname{image} i|_{\mathcal{D}^t}$ is definable, meaning that
$$e_\mathcal{T}(\operatorname{image} i|_{\mathcal{D}^t}) \subset Strings \simeq \mathbb{N}$$
is a set definable by a first order formula in arithmetic. So that in particular the sentence:
$$T \in \mathcal{D}^t$$
is logically equivalent to a first order sentence in arithmetic.

Likewise, if $T \in \operatorname{image} i|_{\mathcal{D}^t}$ then by $T(T,m)$ we mean $i^{-1}(T)(T,m)$, so that in what follows the sentence "T is not T-decided" makes sense for such a T. Explicitly, for $T \in \operatorname{image} i|_{\mathcal{D}^t}$, T is not T-decided, will mean that T is not $i^{-1}(T)$-decided.

Definition 4.1. *For a $D \in \mathcal{D}$, we say that D is* **stably sound on** $T \in \mathcal{T}$ *if*

$$(T \text{ is } D\text{-decided}) \implies (T \in \mathcal{D}^t) \wedge (T \text{ is not } T\text{-decided}).$$

We say that D is **stably sound** *if it is stably sound on all T. We say that D* **stably decides** $\mathcal{P}(T)$ *if:*

$$(T \in \mathcal{D}^t) \wedge (T \text{ is not } T\text{-decided}) \implies T \text{ is } D\text{-decided.}$$

We say that D **stably soundly decides** $\mathcal{P}(T)$ *if D is stably sound on T and stably decides $\mathcal{P}(T)$. We say that D* **stably soundly decides** $\mathcal{P}$ *if D* **stably soundly decides** $\mathcal{P}(T)$ *for all $T \in \mathcal{T}$.*

The informal interpretation of the above is that each $D \in \mathcal{D}$ is understood as an operation with the properties:

- For each T, n $D(T,n) = +$ only if D decides at the moment n that $T \in \mathcal{D}^t$ and T is not T-decided.
- For each T, n $D(T,n) = -$ only if D does not decide/assert at the moment n that $T \in \mathcal{D}^t$, or D does not assert at the moment n that T is not T-decided.

Lemma 4.2. *If D is stably sound on $T \in \mathcal{T}$ then*

$$\neg\Theta_{D,T} \vee \neg(T \text{ is } D\text{-decided}).$$

Proof. If

$$T \text{ is } D\text{-decided}$$

then since D is stably sound on T, T is not T-decided, so of course $\neg\Theta_{D,T}$. □

Theorem 4.3. *There is no (stably) computable $D \in \mathcal{D}$ that stably soundly decides $\mathcal{P}$.*

Proof. Suppose that $D \in \mathcal{D}$ stably soundly decides $\mathcal{P}$ then by the above lemma we obtain:

$$\forall T \in \mathcal{D}^t : \Theta_{D,T} \implies \neg(T \text{ is } D\text{-decided}). \tag{4.4}$$

But it is immediate:

$$\forall T \in \mathcal{D}^t : \Theta_{D,T} \implies (\neg(T \text{ is } D\text{-decided}) \implies \neg(T \text{ is } T\text{-decided})). \tag{4.5}$$

So combining (4.4), (4.5) above we obtain

$$\forall T \in \mathcal{D}^t : \Theta_{D,T} \implies \neg(T \text{ is } T\text{-decided}).$$

But D stably soundly decides $\mathcal{P}$ so we conclude:

$$\forall T \in \mathcal{D}^t : \Theta_{D,T} \implies (T \text{ is } D\text{-decided}).$$

But this is a contradiction to (4.4) unless

$$\forall T \in \mathcal{D}^t : \neg\Theta_{D,T},$$

which is what we wanted to prove. □

We can strengthen the result as follows.

Definition 4.6. *For $D \in \mathcal{D}^t$, we say that $\mathcal{R}(D)$ holds if for any $T \in \mathcal{D}^t$ such T is not T-decided:*

$$\exists T' \in \mathcal{D}^t : (D \text{ stably decides } \mathcal{P}(T')) \wedge (T \simeq_s T').$$

Theorem 4.7. *For $D \in \mathcal{D}$ the following cannot hold simultaneously: D is stably sound, D is (stably) computable and $\mathcal{R}(D)$ holds.*

Proof. Suppose that D is stably computed by some $T \in \mathcal{D}^t$. If D is stably sound then by Lemma 4.2

$$\neg(T \text{ is } D\text{-decided}),$$

and so

$$\neg(T \text{ is } T\text{-decided}),$$

since T stably computes D. Consequently,

$$\mathcal{R}(D) \implies (\exists T' \in \mathcal{D}^t : (T' \simeq^s T) \wedge (T' \text{ is } D\text{-decided})).$$

Combining with Lemma 4.2 we get:

$$\mathcal{R}(D) \implies \exists T' \in \mathcal{D}^t : (T' \simeq^s T) \wedge \neg\Theta_{D,T'},$$

so that if $\mathcal{R}(D)$ then we obtain a contradiction since:

$$(T' \simeq^s T) \wedge \neg\Theta_{D,T'} \implies \neg\Theta_{D,T}.$$

□

Proof of Theorem 1.2. Suppose otherwise that we have such an M, so M is computable, is stably consistent, and the deductive closure C of $M^s(\mathbb{N})$ contains T, for T the set of true sentences for the standard model of arithmetic. Consequently $C = T$ since C is consistent, for given $\alpha \in C$ either $\alpha \in T$ or $\neg\alpha \in T$, and the latter gives that α and $\neg\alpha$ are in C which would contradict consistency, this simple argument was suggested to me by P. Koellner.

Let CM be as in the Lemma 3.8. As already discussed the sentence $T \in \mathcal{D}^t$ is logically equivalent to a first order sentence in arithmetic, likewise $(T \in \mathcal{D}^t) \wedge (T$ is not T-decided) is logically equivalent to a first order sentence in arithmetic, and the translation is computable. Indeed this kind of translation already appears in the original work of Turing [1].

Let then $s(T) \in \mathcal{A}$ be the sentence logically equivalent to

$$(T \in \mathcal{D}^t) \wedge (T \text{ is not } T\text{-decided}).$$

Define an encoded map by $\widetilde{D}_M \in \mathcal{D}$ by

$$\widetilde{D}_M(T, n) := D_M(s(T), n)$$

for D_M defined as in Section 3. By observation above that $M^s = T$ and by the second part of Lemma 3.5 $\widetilde{D}_M$ is stably sound. By first part of Lemma 3.5 $\widetilde{D}_M$ is computable, and $\widetilde{D}_M$ stably decides $\mathcal{P}$ again by assumptions on M and second part of Lemma 3.5. But this contradicts Theorem 4.3. □

5. Incompleteness for stably consistent Turing machines

For each $T \in \mathcal{T}$ let $s(T) \in \mathcal{A}$ be the sentence logically equivalent to:

$$(T \in \mathcal{D}^t) \wedge (T \text{ is not } T\text{-decided}),$$

where this is interpreted as in the previous section.

Proposition 5.1. *Suppose that M is speculative, stably ω-consistent and $M^s \supset RA$. Let CM be as in the Lemma 3.8. Denote by $Z : \mathcal{T} \times \mathbb{N} \to \pm$ the decision map defined by:*

$$Z(T, n) := \widetilde{D}_{CM}(T, n) := D_{CM}(s(T), n),$$

where the latter is as in the previous section. Then

$$\forall T \in \mathcal{T} : \neg\Theta_{Z,T} \vee \neg(M^s \vdash s(T)) \wedge \neg(M^s \vdash \neg s(T)).$$

Moreover, $\Theta_{Z,T} \implies s(T)$ is a theorem of set theory, under standard interpretation of all terms.

Proof. We will be arguing, as in the rest of the paper, within Zermelo-Fraenkel set theory ZF as this where all terms naturally fit and we wish to avoid all meta-logic.

Suppose $\Theta_{Z,T}$. And suppose in addition $M^s \vdash s(T)$. Then by the set theoretic construction, T is Z-decided and so since $\Theta_{Z,T}$, T is T-decided, or more explicitly: there is an m s.t. $T(T, m) = +$ and s.t. there is no $n > m$ s.t. $T(T, m) = -$, translating to arithmetic this is just the sentence $\neg s(T)$. In other words:

$$(\Theta_{Z,T} \wedge M^s \vdash s(T)) \implies \neg s(T),$$

is a theorem of set theory. The sentence $\neg s(T)$ is logically equivalent to a sentence in arithmetic of the form:

$$\exists m : \rho(m) \wedge \forall n\, \gamma(m, n).$$

Set ϕ to be the formula

$$\rho(m) \wedge \forall n : \gamma(m, n).$$

Note that for each m, n: $\rho(m), \gamma(m, n)$ are RA-decidable. So ZF proves:

$$\neg s(T) \implies (\exists m : RA \vdash \rho(m) \wedge \forall n : RA \vdash \gamma(m, n)).$$

In particular:

$$\neg s(T) \implies (\exists m \forall n : M^s \vdash \rho(m) \wedge M^s \vdash \gamma(m, n)).$$

Then by the assumption that M is speculative

$$\neg s(T) \implies \exists m : M^s \vdash \phi(m),$$

so that

$$\neg s(T) \implies M^s \vdash \neg s(T).$$

Thus

$$(\Theta_{Z,T} \wedge M^s \vdash s(T)) \implies \neg Con(M^s),$$

for $Con(M^s)$ expressing consistency of M^s, which would be a contradiction and hence

$$\Theta_{Z,T} \implies \neg M^s \vdash s(T).$$

Now suppose

$$\Theta_{Z,T} \wedge M^s \vdash \neg s(T),$$

then

$$M^s \vdash \exists m : \phi(m),$$

and by ω-consistency

$$\neg(\forall m : M^s \vdash \neg\phi(m)).$$

So for some m_0

$$M^s \vdash \phi(m_0).$$

Since $\rho(m_0)$ is RA-decidable since $M^s \vdash RA$, and since M^s is consistent we have: $RA \vdash \rho(m_0)$. Also

$$\forall n : RA \vdash \gamma(m_0, n),$$

as $\gamma(m_0, n)$ is RA-decidable and otherwise since $M^s \supset RA$ we would again contradict consistency of M^s. Consequently ZF proves:

$$M^s \vdash \neg s(T) \implies RA \vdash \rho(m_0) \wedge \forall n : RA \vdash \gamma(m_0, n),$$

in other words ZF proves:

$$M^s \vdash \neg s(T) \implies \neg s(T),$$

so that T is T-decided. Since $\Theta_{Z,T}$, T is Z-decided, hence by construction of Z, $M^s \vdash s(T)$, so we again obtain that M^s is inconsistent, which is again a contradiction and so

$$\Theta_{Z,T} \implies \neg M^s \vdash \neg s(T)$$

.

Now for the last part of the theorem. By the first part of the above argument, under the assumption that M is speculative, stably ω-consistent and $M^s \supset RA$,

$$\Theta_{Z,T} \implies \neg(M^s \vdash s(T))$$

is a theorem of set theory, under standard interpretation of all terms. But $\neg(M^s \vdash s(T))$ by construction is equivalent to the sentence: T is not T-decided, that is to $s(T)$. So that

$$\Theta_{Z,T} \implies s(T)$$

is a theorem of set theory. □

Proof of Theorem 1.3. Suppose that we have a stably ω-consistent, computable

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

s.t. $M^s \supset RA$. Set

$$N := M_{spec}$$

where the right hand side is as in Theorem 3.11, then N is computable, speculative, ω-consistent, and $N^s \supset M^s$. Then CN is computable and so $\widetilde{D}_{CN}$ is computable by a Turing machine we name T. Then by the proposition above

$$\neg(N^s \vdash s(T)) \wedge \neg(N^s \vdash \neg s(T)),$$

and so

$$\neg(M^s \vdash s(T)) \wedge \neg(M^s \vdash \neg s(T)).$$

The last part of the theorem follows by the last part of the proposition above. □

Acknowledgements. Dennis Sullivan, Bernardo Ameneyro Rodriguez, David Chalmers, and in particular Peter Koellner for helpful discussions on related topics.

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] D. Deutsch, *Quantum theory, the Church-Turing principle and the universal quantum computer.*, Proc. R. Soc. Lond., Ser. A, 400 (1985), pp. 97–117.

[6] S. Feferman, *Are There Absolutely Unsolvable Problems? Gödel's Dichotomy*, Philosophia Mathematica, 14 (2006), pp. 134–152.

[7] C. Fields, D. Hoffman, C. Prakash, and M. Singh, *Conscious agent networks: Formal analysis and application to cognition*, Cognitive Systems Research, 47 (2017).

[8] P. Grindrod, *On human consciousness: A mathematical perspective*, Network Neuroscience, 2 (2018), pp. 23–40.

[9] 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.

[10] R. G. Jeroslow, *Experimental logics and Δ_2^0-theories*, J. Philos. Log., 4 (1975), pp. 253–267.

[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] M. Kasa, *Experimental logics, mechanism and knowable consistency*, Theoria, 78 (2012), pp. 213–224.

[14] A. Kent, *Quanta and qualia*, Foundations of Physics, 48 (2018), pp. 1021–1037.

[15] M. Kikuchi and T. Kurahashi, *Generalizations of Gödel's incompleteness theorems for Σ_n-definable theories of arithmetic*, Rev. Symb. Log., 10 (2017), pp. 603–616.

[16] 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.

[17] ———, *On the question of whether the mind can be mechanized, ii: Penrose's new argument*, Journal of Philosophy, 115 (2018), pp. 453–484.

[18] K. Kremnizer and A. Ranchin, *Integrated information-induced quantum collapse*, Foundations of Physics, 45 (2015), pp. 889–899.

[19] R. Penrose, *Beyond the shadow of a doubt*, Psyche, (1996).

[20] B. Rosser, *Extensions of some theorems of Gödel and Church*, J. Symb. Log., 1 (1936), pp. 87–91.

[21] P. S. Saeed Salehi, *Godel-rosser's incompleteness theorems for non-recursively enumerable theories*, Journal of Logic and Computation, 27-5 (2017).

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