Direct link to author's version

INCOMPLETENESS FOR STABLY COMPUTABLE THEORIES

YASHA SAVELYEV

Abstract. We prove, for stably computably enumerable (stably c.e. for short), theories in the language of arithmetic, direct generalizations of the first and second incompleteness theorems of Gödel. Stably c.e. sets coincide with Σ_2 sets, however the Turing, stably c.e. presentation allows us to make all our theorems explicitly constructive, with proofs using only basic theory of computability, once we categorify the theory Gödel encodings, which appears to be a new approach. In particular, there is no use of the diagonal lemma or other meta-logic. This also gives a diagonal lemma free proof of the original Gödel theorems. As an upshot, our theorems are in principle formalized in set theory ZF, which yields the following set theoretic analogue of the second incompleteness theorem. Let F be a theory in the language of set theory s.t. $F \vdash ZF$. Let $F_{\mathcal{A}}$ be the set of first order sentences of arithmetic provable by F. Suppose that F is consistent, then

$$(F \nvdash F_{\mathcal{A}} \text{ is 1-consistent}) \vee (\forall T : F \nvdash T \text{ stably computes } F_{\mathcal{A}}),$$

where T is a Turing machine. The entirety of the paper is from first principles, with only the "encoding category" requiring some priors. We are motivated by some physical questions, particularly as elaborated by Roger Penrose, which we briefly sketch.

1. Introduction

For an introduction/motivation based around physical ideas the reader may see Appendix A. We begin by quickly introducing the notion of stable computability, in a specific context of theories of arithmetic.

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

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

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

Definition 1.1.

- $\alpha \in \mathcal{A}$ **is** M**-stable** *if there is an* m *with* $M(m) = (\alpha, +)$ *s.t. there is no* $n > m$ *with* $M(n) = (\alpha, -)$. *Let* $M^s \subset \mathcal{A}$ *denote the set of* M*-stable* α*, called* **the stabilization of** M.

Remark 1.2. *For an informal example of how such an* M *may appear in practice consider the following. With* $\mathbb{N}$ *playing the role of time,* M *might be a device producing sentences of arithmetic that it believes to be true, at each moment* $n \in \mathbb{N}$. *But* M *is allowed to also correct itself. In this case:* $M(n) = (\alpha, +)$ *only if at the moment* n M *decides that* α *is true. While* $M(m) = (\alpha, -)$ *only if* $M(n) = (\alpha, +)$ *for some* $n < m$, *and at the moment* m, M *no longer asserts that* α *is true, either because at this moment* M *is unable to decide* α, *or because it has decided it to be false.*

The following definition is preliminary, as we did not yet define (our particularly version of) Turing machines with abstract sets of inputs and outputs, see Section 2.1, and Definition 3.2 for a complete definition.

Definition 1.3. *A subset* $S \subset \mathcal{A}$, *is called* **stably computably enumerable** *or stably c.e. if there is a Turing machine* $T : \mathbb{N} \to \mathcal{A} \times \{\pm\}$ *so that* $S = T^s$. *In this case, we will say that* T **stably computes** S.

It is fairly immediate that a stably computable S is Σ_2 definable. The converse is also true, every Σ_2 definable set is stably computable, to prove this we may build on Example 3.3, to construct an oracle and then use the theorem of Post, (see [14]), relating the arithmetic hierarchy with the theory of Turing degrees. We omit the details.

Let RA denote Robinson arithmetic that is Peano arithmetic PA without induction. Let $\mathcal{F}_0$ denote the set of RA-decidable formulas ϕ in arithmetic with one free variable. In what follows, by a ***theory*** in the language of arithmetic, we just mean a subset $F \subset \mathcal{A}$, with no other conditions.

We recall the following:

Definition 1.4. *Given a theory F in the language of arithmetic, we say that it is* 1**-consistent** *if it is consistent, if $F \vdash RA$, and if for any formula $\phi \in \mathcal{F}_0$ the following holds:*

$$F \vdash \exists m : \phi(m) \implies \neg \forall m : F \vdash \neg \phi(m).$$

We say that it is 2**-consistent** *if the same holds for Π_1 formulas ϕ with one free variable, more specifically formulas $\phi = \forall n : g(m,n)$, with g RA-decidable.*

The first theorem below is a direct generalization of the modern form of Gödel's first incompleteness theorem, as we mainly just weaken the assumption of F being c.e. to being stably c.e.

Theorem 1.5. *Suppose that $F \subset \mathcal{A}$ is stably c.e., and $F \vdash PA$. Then there is an explicitly constructed and elementary from a specification of a Turing machine T stably computing F, sentence $\alpha(F) \in \mathcal{A}$ so that $F \nvdash \alpha(F)$ if F is 1-consistent and $F \nvdash \neg\alpha(F)$ if F is 2-consistent.*

The second theorem below directly generalizes the more striking second incompleteness theorem of Gödel. Some ramifications of this are discussed in Appendix A.

Theorem 1.6. *Let F, T and $\alpha(F)$ be as in the theorem just above. Then*

$$(F \text{ is 1-consistent}) \implies \alpha(F) \tag{1.7}$$

is a theorem of PA. More formally stated, the sentence (1.7) is equivalent in ZF to an arithmetic sentence provable by PA. In particular, $\alpha(F)$ is true in the standard model of arithmetic whenever F is 1-consistent. Consequently, since by assumption $F \vdash PA$, if F is 1-consistent then:

$$F \nvdash (F \text{ is 1-consistent}),$$

that is F cannot prove its own 1-consistency.

As pointed out in [13] there are examples of complete, consistent Δ_2 definable extensions of PA. So that stronger assumptions like 1-consistency are necessary.

Corollary 1.8. *Suppose $F \subset \mathcal{A}$ and $F \vdash PA$ then either F is not stably c.e. or F is not 1-consistent, or there exists an explicitly constructed and elementary (given T stably computing F) true sentence $G(F)$ s.t. $F \nvdash G(F)$. If F is in addition 2-consistent then $G(F)$ is F-independent.*

The above theorems will be entirely formalized in set theory ZF. In particular, we obtain the following partial corollary:

Theorem 1.9. *Let F be a theory in the language of set theory s.t. $F \vdash ZF$. Let $F_{\mathcal{A}}$ be the set of first order sentences of arithmetic provable by F. Then if F is consistent:*

$$(F \nvdash F_{\mathcal{A}} \text{ is 1-consistent}) \vee (\forall T : F \nvdash T \text{ stably computes } F_{\mathcal{A}}), \tag{1.10}$$

where T is a Turing machine as in Definition 1.3.

In a sense, this extends the following:

Theorem 1.11. *(Corollary of M. Kikuchi and T. Kurahashi* [7, Proposition 5.3, Theorem 5.6]*) Let F be a theory in the language of arithmetic s.t. $F \vdash PA$. Then if F has a Σ_2 definition σ, and F is 1-consistent, there is a particular arithmetic sentence $Con_{\sigma,1}(F)$ entailing 1-consistency of F s.t.*

$$F \nvdash Con_{\sigma,1}(F). \tag{1.12}$$

One distinction with Theorem 1.9 is that the consistency sentence $Con_{\sigma,1}(F)$ must be specifically constructed from the definition σ.

1.1. **Relationship with known results.** The main distinction with the theorems of Gödel is that the set F is merely Σ_2 definable when F is stably c.e. As far as the first incompleteness Theorem 1.5, there is much previous history on attempts of generalizations of a similar kind dating almost back to Gödel. To give one recent example, in the work of Salehi and Seraji [13], which we also recommend for additional references, the general Corollary 2.11 in part implies Theorem 1.5. The main difference is that our sentence is explicitly constructed using the language of Turing machines and is very elementary. In particular, we completely avoid the diagonal Lemma, and our overall methods are very different.

Surprisingly, the authors of [13] point out that in general, when n is at least 3, for a Σ_{n+1} definable, n-consistent theory F there is no constructible, Π_{n+1} definable, F-independent sentence, although such a sentence does exist.

On the other hand, I am not aware of direct generalizations of the second incompleteness theorem of Gödel in the form of Theorem 1.6. But Theorem 1.11 of M. Kikuchi and T. Kurahashi is one generalization. Once again our methods are totally different, the methods of this paper being based essentially on recursion theory, while the methods of 1.11 based on the diagonal Lemma in logic. This makes it difficult to a non-expert like the author to give a more in depth comparison of these results.

1.2. **Generalizations to Σ_n.** There are natural candidates for how to generalize theorems above. We may replace $M : \mathbb{N} \to \mathcal{A} \times \{\pm\}$ by $M : \mathbb{N}^n \to \mathcal{A} \times \{\pm\}$, using this we can define a notion of n-stable computability, specializing to stable computability for $n = 1$. In terms of arithmetic complexity this is exactly the class Σ_{n+1}. We leave this for future developments.

1.3. **The original incompleteness theorems.** Finally, we note that, our argument also readily reproves the original first and second incompleteness theorems of Gödel from first principles and within set theory [1], with our "Gödel sentence" constructed directly via the theory of Turing machines. In particular, the somewhat mysterious to non-logicians diagonal Lemma, ordinarily used in modern renditions of the proof, is not used.

2. Some preliminaries

For more details on Turing machines we recommend the book of Soare [14]. Our approach here is however possibly novel in that we do not work with concrete Gödel encodings, instead abstractly axiomatizing their expected properties. This results in a certain encoding category, which will allow us to work with the language of set theory more transparently. This setup will be necessary for our constructive approach to the generalized incompleteness theorems.

We go over some basics primarily to set notation.

Definition 2.1. *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 $\mathfrak{s}$, $\delta^M(\mathfrak{s})$ will denote the successor state of $\mathfrak{s}$, obtained by executing the instructions set of M on $\mathfrak{s}$, or in other words $\delta^M(\mathfrak{s})$ is one step forward from $\mathfrak{s}$.*

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 2.2. *A* **Turing computation**, *or* **computation sequence** *for M is a possibly not eventually constant sequence*

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

of total states of M, determined by the input Σ and M, with $\mathfrak{s}_0$ suitable initial configuration, in particular having internal state is q_0, and where $\mathfrak{s}_{i+1} = \delta^M(\mathfrak{s}_i)$. If the sequence $\{\mathfrak{s}_i\}_{i=0}^{i=\infty}$ is eventually constant: $\mathfrak{s}_i = \mathfrak{s}_\infty$ for $\forall i > n$, for some n, and if the internal state of $\mathfrak{s}_\infty$ is a final state, then we say that the computation **halts**. *For a given Turing computation $*M(\Sigma)$, we will write*

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

[1] Assuming existence of a certain encoding category $\mathcal{S}$, which is in part classical.

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. 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 $q_{reject} \in F$ so that $T(\Sigma)$ halts with internal state q_{reject}. In this case we say that T ***rejects*** Σ. The set $\mathcal{I} \subset Strings$ of strings not rejected by T 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 2.3. *We denote by $\mathcal{T}$ the set of all Turing machines with a distinguished final machine state q_{reject}.*

Instead of tracking q_{reject} explicitly, we may 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 2.4. *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 $\mathcal{I}$. In other words the set of permissible strings of T is $\mathcal{I}$, and as a partial function on $\mathcal{I}$, $T = f$. We say that f is* **computable** *if such a T exists.*

2.1. **Abstractly encoded sets and abstract Turing machines.** The material of this section will be used in the main arguments. Instead of specifying Gödel encodings we just axiomatize their expected properties. Working with encoded sets/maps as opposed to concrete subsets of $Strings$/functions will have some advantages as we can involve set theory more transparently, and construct computable maps axiomatically. This kind of approach is likely obvious to experts, but I am not aware of this being explicitly introduced in computability theory literature.

An ***encoding*** of a set A is at the moment just an injective set map $e : A \to Strings$. But we will need to axiomatize this further. The ***encoding category*** $\mathcal{S}$ will be a certain 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. More explicitly, the set of objects of $\mathcal{S}$ consists of some set of pairs (A, e_A) where A is a set, and $e_A : A \to Strings$ an embedding, determined by A. We may denote $e_A(A)$ by A_e. We now describe the morphisms. Suppose that we are given a commutative diagram:

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

where T is a partial map, and $T_e \in \mathcal{T}$ is a Turing machine in the standard sense above, with the set of permissible inputs A_e. Then the set of morphisms from (A, e_A) to (B, e_B) consists of equivalence classes of such commutative diagrams (T, T_e), where the equivalence relation is $(T, T_e) \sim (T', T'_e)$ if $T = T'$. If in addition a morphism has a representative (of the equivalence class) (T, T_e), with T_e primitive recursive then we call it a ***a primitive recursive morphism***.

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

We call such an $A \in \mathcal{S}$ an ***abstractly encoded set*** so that $\mathcal{S}$ is a category of abstractly encoded sets. The morphisms set from between objects A, B in $\mathcal{S}$ as usual will be denoted by $hom_{\mathcal{S}}(A, B)$. The composition maps

$$hom_{\mathcal{S}}(A, B) \times hom_{\mathcal{S}}(B, C) \to hom_{\mathcal{S}}(A, C)$$

are defined once we fix a prescription for the composition of Turing machines. That is

$$[(T, T_e)] \circ [(T', T'_e)] = [(T \circ T', T_e \circ T'_e)],$$

for $[\cdot]$ denoting the equivalence class and $[(T, T_e)] \in hom_{\mathcal{S}}(B, C)$ and $[(T', T'_e)] \in hom_{\mathcal{S}}(A, B)$.

In addition, we ask that $\mathcal{S}$ satisfies the following axioms.

(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 computable partial function $Strings \to Strings$ with range S.

(2) For $A, B \in \mathcal{S}$,
$$(A_e \cap B_e) = \emptyset.$$
In particular each $A \in \mathcal{S}$ is determined by A_e. (This particular axiom is not used in an essential way, by it does simplify the discussion.)

(3) 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 $\mathcal{S}$, so that in particular we have a commutative diagram:
$$\begin{array}{ccc} A \times B & \xrightarrow{pr^A} & A \\ \downarrow{\scriptstyle e_{A\times B}} & & \downarrow{\scriptstyle e_A} \\ (A \times B)_e & \xrightarrow{pr^A_e} & A_e, \end{array}$$
similarly for pr^B.

(4) If $f : A \to B$ completes to a morphism of $\mathcal{S}$, and $g : A \to C$ completes to a morphism of $\mathcal{S}$ then $A \to B \times C$, $a \mapsto (f(a), g(a))$ completes to a morphism of $\mathcal{S}$. This combined with Axiom 3 implies that if $f : A \to B$, $g : C \to D$ extend to morphisms of $\mathcal{S}$ then the map $A \times B \to C \times D$, $(a, b) \mapsto (f(a), g(b))$ extends to a morphism of $\mathcal{S}$.

(5) The set $\mathcal{U} = Strings$ and $\mathcal{T}$ are encoded i.e. $\mathcal{U}, \mathcal{T} \in \mathcal{S}$. (We use the alternative name $\mathcal{U}$ for $Strings$, as in this case the encoding map has the same domain and range, which is possibly confusing.) The partial map
$$U : \mathcal{T} \times \mathcal{U} \to \mathcal{U},$$
$U(T, \Sigma) := T(\Sigma)$ whenever $*T(\Sigma)$ halts and undefined otherwise, extends to a morphism of $\mathcal{S}$. We can understand a representative, (U, U_e), of the morphism, as the "universal Turing machine".

(6) The encoding map $e_{\mathcal{U}} : \mathcal{U} \to Strings$ is classically computable. (This makes sense since $\mathcal{U} = Strings$). It immediately follows from this that for $A \in \mathcal{S}$ the encoding map $e_A : A \to \mathcal{U}$ itself extends to a morphism of $\mathcal{S}$.

(7) The next axiom gives a prescription for construction of Turing machines. Let $A, B, C \in \mathcal{S}$, and suppose that $f : A \times B \to C$ extends to a morphism of $\mathcal{S}$. Let $f^a : B \to C$ be the map $f^a(b) = f(a, b)$. Then there is a map
$$s : A \to \mathcal{T}$$
so that for each a $(f^a, s(a))$ represents a morphism $B \to C$, and so that s extends to a morphism of $\mathcal{S}$.

(8) The final axiom is for utility. If $A \in \mathcal{S}$ then $L(A) \in \mathcal{S}$, where
$$L(A) = \bigcup_{n \in \mathbb{N}} Maps(\{0, \ldots, n\}, A),$$
and $Maps(\{0, \ldots, n\}, A)$ denotes the set of total maps. We also have:
   (a) $\mathbb{N}$ is encoded.

(b) Let $A \in \mathcal{S}$ and let
$$\text{length} : L(A) \to \mathbb{N},$$
be the length function, s.t. for $l \in L(\mathcal{A})$, $l : \{0, \dots n\} \to A$, $\text{length}(l) = n$. Then length extends to a morphism of $\mathcal{S}$.

(c)
$$P : L(A) \times \mathbb{N} \to A,$$
extends to a morphism of $\mathcal{S}$, where $P(l, i) := l(i)$, or undefined for $i > \text{length}(l)$.

(d) For $A, B \in \mathcal{S}$ and $f : A \to L(B)$ a partial map, suppose that:
- The partial map $A \times \mathbb{N} \to B$, $(a, n) \mapsto P(f(a), n)$ extends to a morphism of $\mathcal{S}$.
- The partial map $A \to \mathbb{N}$, $a \mapsto \text{length}(f(a))$ extends to a morphism of $\mathcal{S}$.

Then f extends to a morphism of $\mathcal{S}$.

Lemma 2.5. *If $f : A \to B$ extends to a morphism of $\mathcal{S}$ then the map $L(f) : L(A) \to L(B)$,*
$$l \mapsto \begin{cases} i \mapsto f(l(i)), & \text{if } f(l(i)) \text{ is defined for all } 0 \leq i \leq \text{length}(l) \\ undefined, & \text{otherwise}, \end{cases}$$
extends to a morphism of $\mathcal{S}$. Also, the map $LU : \mathcal{T} \times L(\mathcal{U}) \to L(\mathcal{U})$,
$$l \mapsto \begin{cases} i \mapsto U(T, (l(i))), & \text{if } U(T, (l(i))) \text{ is defined for all } 0 \leq i \leq \text{length}(l) \\ undefined, & \text{otherwise} \end{cases}$$
extends to a morphism of $\mathcal{S}$.

Proof. This is just a straightforward application of the axioms and Axiom 8 in particular. We leave the details as an exercise. □

The above axioms suffice for our purposes, but there are a number of possible extensions (dealing with other set theoretic constructions like the set theoretic sum). 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 $\mathbb{N}, \mathcal{T}, \mathcal{A}, \{\pm\}$, with $\{\pm\}$ a set with two elements $+, -$. The encoding of $\mathbb{N}$ should be suitably natural so that for example the map
$$\mathbb{N} \to \mathbb{N}, \quad n \mapsto n + 1$$
completes to a primitive recursive morphism in $\mathcal{S}$. Strictly speaking this is part of the axioms. The main naturality properties for the encoding of $\mathcal{T}$ are already stated as Axioms 5, 7. The naturality axioms for $\mathcal{A}$ will be implicitly specified further on as needed.

The fact that such encoding categories $\mathcal{S}$ exist is a folklore theorem starting with foundational work of Gödel, Turing and others. Indeed, $\mathcal{S}$ can be readily constructed from standard Gödel type encodings. For example Axiom 7, in classical terms, just reformulates the following elementary fact, which follows by the "s-m-n theorem" Soare [14, Theorem 1.5.5]. Given a classical 2-input Turing machine
$$T : Strings \times Strings \to Strings,$$
there is a Turing machine $s_T : Strings \to Strings$ s.t. for each Σ $s_T(\Sigma)$ is the Turing-Gödel encoding string (a.k.a. program) of a Turing machine computing the map $\rho \mapsto T(\Sigma, \rho)$.

In modern terms, the construction of $\mathcal{S}$ is essentially a part of a definition of a computer programming language (with algebraic data types, e.g. Haskell.)

Definition 2.6. *For $A, B \in \mathcal{S}$, an* **abstract Turing machine from A to B** *will be a synonym for a class representative of an element of $hom_{\mathcal{S}}(A, B)$. In other words it is a pair (T, T_e) as above, with $T : A \to B$. We will say that the corresponding Turing machine T_e* **encodes** *the map T or is the encoding of T. We say that (T, T_e) is* **total** *when T is total. We may simply write $T : A \to B$ for an abstract Turing machine, when T_e is implicit. We write $\overline{hom}_{\mathcal{S}}(A, B)$ for the set of abstract Turing machines from A to B.*

Often we will say Turing machine in place of abstract Turing machine, since usually there can be no confusion as to the meaning.

We define $\mathcal{T}_{\mathcal{S}}$ to be the set of abstract Turing machines relative to $\mathcal{S}$ as above. More formally:

$$\mathcal{T}_{\mathcal{S}} = \bigcup_{(A,B)\in\mathcal{S}^2} \overline{hom}_{\mathcal{S}}(A,B).$$

$\mathcal{T}_{\mathcal{S}}$ will not be encoded.

We write $Maps(A,B)$ for the set of partial maps from A to B, and if we say map this just means partial map, unless we say total map. Let

$$\mathcal{M}_{\mathcal{S}} := \bigcup_{(A,B)\in\mathcal{S}^2} Maps(A,B).$$

Given a Turing machine $T : A \to B$, we have an associated map $fog(T) : A \to B$ defined by forgetting the additional structure T_e. However, we may also just write T for this map by abuse of notation. So we have a forgetful map

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

which forgets the extra structure of an encoding Turing machine.

Definition 2.7. *We say that $T \in \mathcal{T}_{\mathcal{S}}$* **computes** *$M \in \mathcal{M}_{\mathcal{S}}$ if $fog(T) = M$. We say that M is* **computable** *if some T computes M, or equivalently if M extends to a morphism of $\mathcal{S}$.*

3. Stable computability and arithmetic

In this section, general sets, often denoted as B, are intended to be encoded. And all maps are partial maps, unless specified otherwise.

Definition 3.1. *Given a Turing machine or just a map:*

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

We say that $b \in B$ **is M-stable** *if there is an m with $M(m) = (b,+)$ and there is no $n > m$ with $M(n) = (b,-)$.*

Definition 3.2. *Given a Turing machine or just a map*

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

we define

$$M^s \subset B$$

to be the set of all the M-stable b. We call this the **stabilization of** *M. We say that $S \subset B$ is* **stably c.e.** *if $S = M^s$ for M as above.*

In general M^s may not be computable even if M is computable. Explicit examples of this sort can be readily constructed as shown in the following.

Example 3.3. Let Pol denote the set of all Diophantine polynomials, (also abstractly encoded). We can construct a total computable map

$$A : \mathbb{N} \to Pol \times \{\pm\}$$

whose stabilization consists of all Diophantine (integer coefficients) polynomials with no integer roots. Similarly, we can construct a computable map D whose stabilization consists of 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 the case of Diophantine polynomials, here is an (inefficient) example. Let

$$Z : \mathbb{N} \to Pol, N : \mathbb{N} \to \mathbb{Z}$$

be total bijective computable maps. The encoding of Pol should be suitably natural so that in particular the map

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

is computable. In what follows, for each $n \in \mathbb{N}$, $A_n \in L(Pol \times \{\pm\})$. $\cup$ will be here and elsewhere in the paper the natural list union operation. More specifically, if $l_1 : \{0, \dots, n\} \to B, l_2 : \{0, \dots, m\} \to B$ are two lists then $l_1 \cup l_2$ is defined by:

$$l_1 \cup l_2(i) = \begin{cases} l_1(i), & \text{if } i \in \{0, \dots, n\} \\ l_2(i), & \text{if } i \in \{n+1, \dots, n+m+1\} \end{cases}. \tag{3.4}$$

If $B \in \mathcal{S}$, it is easy to see by the axioms of $\mathcal{S}$ that $\cup : L(B) \times L(B) \to L(B)$, $(l, l') \mapsto l \cup l'$ is computable.

For $n \in \mathbb{N}$ define A_n recursively by: $A_0 := \emptyset$,

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

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

Note that

$$\forall n \in \mathbb{N} : A_{n+1}|_{\text{domain } A_n} = A_n, \text{ and } \text{length}(A_{n+1}) > \text{length}(A_n),$$

so we may define $A(n) := A_{n+1}(n)$. With this definition $A(\mathbb{N}) = \cup_{n \in \mathbb{N}} \text{image}(A_n)$.

Since E is computable, utilizing the axioms it can be explicitly verified that A is computable, i.e. an encoding Turing machine can be explicitly constructed, from the recursive program above. Moreover, by construction the stabilization A^s consists of all Diophantine polynomials that have no integer roots.

3.1. **Decision maps.** By a ***decision map***, we mean a map of the form:

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

This kind of maps will play a role in our arithmetic incompleteness theorems, and we now develop some of their theory.

Definition 3.5. *Let $B \in \mathcal{S}$, define*

$$A\mathcal{D}_B := \overline{hom}_{\mathcal{S}}(B \times \mathbb{N}, \{\pm\}),$$

and define

$$\mathcal{D}_B := \{T \in \mathcal{T} | T = T'_e \text{ for } (T', T'_e) \in A\mathcal{D}_B\}.$$

Since T' above is uniquely determined, from now on, for $T \in \mathcal{D}_B$, when we write T' it is meant to be of the form above.

First we will explain construction of elements of $\mathcal{D}_B$ from Turing machines of the following form.

Definition 3.6. *Let $B \in \mathcal{S}$, define*

$$A\mathcal{T}_B := \overline{hom}_{\mathcal{S}}(\mathbb{N}, B \times \{\pm\}),$$

and define

$$\mathcal{T}_B := \{T \in \mathcal{T} | T = T'_e \text{ for } (T', T'_e) \in A\mathcal{T}_B\}.$$

From now on, given $T \in \mathcal{T}_{\mathcal{B}}$, if we write T' then it is will be assumed to be of the form above.

Lemma 3.7. *Let $B \in \mathcal{S}$. There is a computable total map*

$$K_B : \mathcal{T} \to \mathcal{T},$$

with the properties:

1. *For each T, $K_B(T) \in \mathcal{T}_B$.*
2. *If $T \in \mathcal{T}_B$ then $K_B(T)$ and T encode the same map $\mathbb{N} \to B \times \{\pm\}$, in other words $T' = (K_B(T))'$.*

Proof. Let $G : \mathcal{T} \times \mathbb{N} \to B \times \{\pm\}$ be the composition of the sequence of maps

$$\mathcal{T} \times \mathbb{N} \xrightarrow{id \times e_{\mathbb{N}}} \mathcal{T} \times \mathcal{U} \xrightarrow{U} \mathcal{U} \xrightarrow{e^{-1}_{B\times\{\pm\}}} B \times \{\pm\},$$

where the last map $e^{-1}_{B\times\{\pm\}}$ is defined by:

$$\Sigma \mapsto \begin{cases} e^{-1}_{B\times\{\pm\}}(\Sigma), & \text{if } \Sigma \in (B \times \{\pm\})_e \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

In particular, this last map is computable as $(B \times \{\pm\})_e$ is by assumption computable/decidable. Hence, G is a composition of computable maps and so is computable. By Axiom 7 there is an induced computable map $K_B : \mathcal{T} \to \mathcal{T}$ so that $K_B(T)$ is the encoding of $G^T : \mathbb{N} \to B \times \{\pm\}$, $G^T(n) = G(T, n)$. By construction, if $T \in \mathcal{T}_B$ then $T' = (K_B(T))'$, so that we are done. □

3.1.1. *Constructing decision Turing machines.*

Definition 3.8. *Let $l \in L(B \times \{\pm\})$. Define $b \in B$ to be l-**stable** if there is an $m \leq \text{length}(l)$ s.t. $l(m) = (b, +)$ and there is no $m < k \leq \text{length}(l)$ s.t. $l(k) = (b, -)$.*

Define

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

to be the map:

$$G(b, T, n) = \begin{cases} \text{undefined}, & \text{if } (K_{\mathcal{A}}(T))'(i) \text{ is undefined for some } 0 \leq i \leq n \\ +, & b \text{ is } l\text{-stable for } l = \{(K_{\mathcal{A}}(T))'(0), \ldots, K_{\mathcal{A}}(T))'(n)\} \\ -, & \text{otherwise.} \end{cases}$$

Let

$$g : \mathbb{N} \to L(\mathbb{N}) \tag{3.9}$$

be the map $g(n) = \{0, \ldots, n\}$, it is clearly computable directly by the Axiom 8. We can express G as the composition of the sequence of maps:

$$B \times \mathcal{T} \times \mathbb{N} \xrightarrow{id \times K_B \times g} B \times \mathcal{T} \times L(\mathbb{N}) \xrightarrow{id \times id \times L(e_{\mathbb{N}})} B \times \mathcal{T} \times L(\mathcal{U}) \xrightarrow{id \times LU} B \times L(\mathcal{U})$$
$$\xrightarrow{id \times L(e^{-1}_{B\times\{\pm\}})} B \times L(B \times \{\pm\}) \to \{\pm\},$$

where the last map is:

$$(b, l) \mapsto \begin{cases} +, & \text{if } b \text{ is } l\text{-stable} \\ -, & \text{otherwise,} \end{cases}$$

which is computable by explicit verification, utilizing the axioms. And where $L(e_{\mathbb{N}}), L(e^{-1}_{B\times\{\pm\}})$ and LU are as in Lemma 2.5. In particular all the maps in the composition are computable and so G is computable.

Let

$$Dec_B : \mathcal{T} \to \mathcal{T}, \tag{3.10}$$

be the computable map corresponding G via Axiom 7, so that $Dec_B(T)$ is the encoding of

$$G^T : B \times \mathbb{N} \to \{\pm\}, G^T(b, n) = G(b, T, n).$$

The following is immediate:

Lemma 3.11. *$Dec_B(T)$ has the property:*

$$\forall T \in \mathcal{T} : Dec_B(T) \in \mathcal{D}_B.$$

Furthermore, if $T \in \mathcal{T}_B$ and is total then $Dec_B(T)$ is total.

Definition 3.12. *For a Turing machine or just a map $D : B \times N \to \{\pm\}$, we say that $b \in B$ is* **D-decided** *if there is an m s.t. $D(b,m) = +$ and for all $n \geq m$ $D(b,n) \neq -$. Likewise, for $T \in \mathcal{D}_B$ we say that $b \in B$ is* **T-decided** *if it is T'-decided. Also for $T \in \mathcal{T}_{\mathcal{A}}$ we say that b is* **T-stable** *if it is T'-stable in the previous sense.*

Lemma 3.13. *Suppose that $T \in \mathcal{T}_B$ and T' is total then b is T-stable iff b is $Dec_B(T)$-decided.*

Proof. Suppose that b is T-stable. In particular, there is an $m \in \mathbb{N}$ so that b is l-stable for $l = \{T'(0), \dots T'(n)\}$ all $n \geq m$. Thus, by construction

$$\forall n \geq m : G(B, T, n) = +,$$

and so b is G^T-decided (this is as above), and so $Dec_B(T)$-decided.

Similarly, suppose that b is $Dec_B(T)$-decided, then there is an m s.t. $G(b,T,m) = +$ and there is no $n > m$ s.t. $G(b,T,n) = -$. It follows, since $T' = (K_B(T))'$, that $\exists m' \leq m : T'(m') = (b,+)$ and there is no $n > m'$ s.t. $T'(n) = (b,-)$. And so b is T-stable.

□

Example 3.14. By the Example 3.3 above there is a Turing machine

$$P = Dec_{\mathcal{A}}(A) : Pol \times \mathbb{N} \to \{\pm\}$$

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

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

Likewise, there is a Turing decision machine that stably soundly decides the halting problem, in this sense.

Definition 3.15. *Given a map*

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

and a Turing machine

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

we say that M' **stably computes** *M if*

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

If $T \in \mathcal{D}_B$ then we say that T stable computes M iff T' stably computes M. Here, as before, $T' \in A\mathcal{D}_B$ is such that $T'_e = T$.

3.2. **Arithmetic decision maps.** Let $\mathcal{A}$ be as in the introduction the set of sentences of arithmetic. Let $\mathcal{T}_{\mathcal{A}}$ be as in Definition 3.6 with respect to $B = \mathcal{A}$. That is elements of $\mathcal{T}_{\mathcal{A}}$ are of the form $T = T'_e$ for $(T', T'_e) \in A\mathcal{T}_{\mathcal{A}} = \overline{hom}_{\mathcal{S}}(\mathbb{N}, \mathcal{A} \times \{\pm\})$.

The following is a version for stably c.e. theories of the classical fact, going back to at least Gödel, that for a theory with a c.e. set of axioms we may computably enumerate its theorems. Moreover, the procedure to obtain the corresponding Turing machine is constructive.

Notation 2. Note that each $T \in \mathcal{T}_{\mathcal{A}}$, determines the set

$$(T')^s \subset \mathcal{A},$$

called the stabilization of T', we hereby abbreviate the notation for this set as T^s.

Lemma 3.16. *There is a computable total map:*

$$C : \mathcal{T} \to \mathcal{T}$$

so that $\forall T \in \mathcal{T} : C(T) \in \mathcal{T}_{\mathcal{A}}$. If in addition $T \in \mathcal{T}_{\mathcal{A}}$ and T' is total then $(C(T))^s$ is the deductive closure of T^s.

Proof. Let $L(\mathcal{A})$ be as in axioms of $\mathcal{S}$, defined with respect to $B = \mathcal{A}$. The following lemma is classical and its proof is omitted. Strictly speaking we of course need that the encoding of $\mathcal{A}$ is suitably natural. We may assume the standard Gödel encoding.

Lemma 3.17. *There is a total computable map:*

$$\Phi : L(\mathcal{A}) \times \mathbb{N} \to \mathcal{A}$$

with the following property. For each $l \in L(\mathcal{A})$, $\Phi(\{l\} \times \mathbb{N})$ is the set of all sentences provable by the theory l, the latter being shorthand for the image of the corresponding map $l : \{0, \dots, n\} \to \mathcal{A}$.

Let $K_{\mathcal{A}}$ be as in Lemma 3.7, with respect to $B = \mathcal{A}$. Define a map

$$\zeta : \mathcal{T} \times \mathbb{N} \times L(\mathcal{A}) \to \{\pm\}$$

by

$$\zeta(T, n, l) = \begin{cases} \text{undefined}, & \text{if } (K_{\mathcal{A}}(T))'(i) \text{ is undefined for some } 0 \leq i \leq n \\ +, & \text{if for each } 0 \leq i \leq n,\ l(i) \text{ is } l\text{-stable for } l = \{(K_{\mathcal{A}}(T))'(0), \dots, K_{\mathcal{A}}(T))'(n)\} \\ -, & \text{otherwise.} \end{cases}$$

We can express ζ as the composition of the sequence of maps

$$\mathcal{T} \times \mathbb{N} \times L(\mathcal{A}) \xrightarrow{K_{\mathcal{A}} \times id \times id} \mathcal{T} \times \mathbb{N} \times L(\mathcal{A}) \xrightarrow{id \times g \times id} \mathcal{T} \times L(\mathbb{N}) \times L(\mathcal{A})$$

$$\xrightarrow{id \times L(e_{\mathbb{N}}) \times id} \mathcal{T} \times L(\mathcal{U}) \times L(\mathcal{A}) \xrightarrow{LU \times id} L(\mathcal{U}) \times L(\mathcal{A}) \xrightarrow{L(e^{-1}_{\mathcal{A} \times \{\pm\}}) \times id} L(\mathcal{A} \times \{\pm\}) \times L(\mathcal{A}) \to \{\pm\}.$$

Here the last map is

$$(l, l') \mapsto \begin{cases} +, & \text{if } l'(i) \text{ is } l\text{-stable, for each } 0 \leq i \leq \text{length}(l') \\ -, & \text{otherwise} \end{cases},$$

it is computable by explicit verification utilizing the axioms. The map g in the second map is as in (3.9). Thus, all maps in the composition are computable and ζ is computable.

Now define G to be the composition of the sequence of maps:

$$\mathcal{T} \times L(\mathbb{N}) \xrightarrow{K_{\mathcal{A}} \times L(e_{\mathbb{N}})} \mathcal{T} \times L(\mathcal{U}) \xrightarrow{LU} L(\mathcal{U}) \xrightarrow{L(e^{-1}_{\mathcal{A} \times \{\pm\}})} L(\mathcal{A} \times \{\pm\}) \xrightarrow{L(pr_{\mathcal{A}})} L(\mathcal{A}),$$

where $pr_{\mathcal{A}} : \mathcal{A} \times \{\pm\} \to \mathcal{A}$ is the natural projection. The third and fourth map are as in Lemma 2.5 for $e^{-1}_{\mathcal{A} \times \{\pm\}}$, as in Lemma 3.7. All the maps in the composition are computable directly by the axioms of $\mathcal{S}$ and so G is computable.

We may now construct our map C. In what follows $\cup$ will be the natural list union operation as previously in (3.4). Set

$$L_n(\mathbb{N}) := \{l \in L(\mathbb{N}) | \max l \leq n,\ \max l \text{ the maximum of } l \text{ as a map}\}.$$

For $n \in \mathbb{N}$, define $U^T_n \in L(\mathcal{A} \times \{\pm\})$ recursively by $U^T_0 := \emptyset$,

$$U^T_{n+1} := U^T_n \cup \bigcup_{l \in L_{n+1}(\mathbb{N})} (\Phi(G(T, l), n+1), \zeta(T, n+1, G(T, l))).$$

As in Example 3.3 we define $U^T : \mathbb{N} \to \mathcal{A} \times \{\pm\}$ by $U^T(n) := U^T_{n+1}(n)$, note that the right-hand side may be undefined since G is only a partial map. So U^T is a partial map. And this induces a partial map

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

$U(T, n) := U^T(n)$. U is computable by explicit verification, utilizing the axioms of $\mathcal{S}$, i.e. an encoding Turing machine can be readily constructed from the recursive program for $\{U^T_n\}$ above. Hence, by the Axiom 7 there is an induced by U computable map:

$$C : \mathcal{T} \to \mathcal{T},$$

s.t. for each $T \in \mathcal{T}$ $C(T)$ computes U^T. If $T \in \mathcal{T}_{\mathcal{A}}$ and is total then $(U^T)^s$ is by construction the deductive closure of $(K_{\mathcal{A}}(T))^s = T^s$. So the map C has the needed property, and we are done. $\square$

Definition 3.18. *Let $\mathcal{F}_0$, as in the introduction, denote the set of formulas ϕ of arithmetic with one free variable so that $\phi(n)$ is an RA-decidable sentence for each n. Let $M : \mathbb{N} \to \mathcal{A} \times \{\pm\}$ be a map (or a Turing machine). The notation $M \vdash \alpha$ will be short for $M^s \vdash \alpha$. We say that M is* **speculative** *if the following holds. Let $\phi \in \mathcal{F}_0$, and set*

$$\alpha_\phi = \forall m : \phi(m), \tag{3.19}$$

then

$$\forall m : RA \vdash \phi(m) \implies M \vdash \alpha_\phi.$$

Note that of course the left-hand side is not the same as $RA \vdash \alpha_\phi$.

We may informally interpret this condition as saying that M initially outputs α as a hypothesis, and removes α from its list (that is α will not be in M^s) only if for some m, $RA \vdash \neg\phi(m)$. Note that we previously constructed an Example 3.3 of a Turing machine, with an analogue of this speculative property. Moreover, we have the following crucial result, which to paraphrase states that there is an operation *Spec* that converts a stably c.e. theory to a speculative stably c.e. theory, at a certain loss of consistency.

Theorem 3.20. *There is a computable total map $Spec : \mathcal{T} \to \mathcal{T}$, with the following properties:*

1. image $Spec \subset \mathcal{T}_{\mathcal{A}}$.
2. *Let $T \in \mathcal{T}_{\mathcal{A}}$. Set $T_{spec} = Spec(T)$ then T'_{spec} is speculative, moreover if T' is total then so is T'_{spec}.*
3. *Using Notation 2, if $T \in \mathcal{T}_{\mathcal{A}}$ then $T^s_{spec} \supset T^s$*
4. *If $T \in \mathcal{T}_{\mathcal{A}}$ and T^s is 1-consistent then T^s_{spec} is consistent.*

Proof. $\mathcal{F}_0, \mathcal{A}$ are assumed to be encoded so that the map

$$ev : \mathcal{F}_0 \times \mathbb{N} \to \mathcal{A}, \quad (\phi, m) \mapsto \phi(m)$$

is computable.

Lemma 3.21. *There is a total computable map $F : \mathbb{N} \to \mathcal{F}_0 \times \{\pm\}$ with the property:*

$$F^s = G := \{\phi \in \mathcal{F}_0 \,|\, \forall m : RA \vdash \phi(m)\}.$$

Proof. The construction is analogous to the construction in the Example 3.3 above. Fix any total, bijective, Turing machine

$$Z : \mathbb{N} \to \mathcal{F}_0.$$

For a $\phi \in \mathcal{F}_0$ we will say that it is n-***decided*** if

$$\forall m \in \{0, \dots, n\} : RA \vdash \phi(m).$$

In what follows each F_n has the type of ordered finite list of elements of $\mathcal{F}_0 \times \{\pm\}$, and $\cup$ will be the natural list union operation, as previously. Define $\{F_n\}_{n\in\mathbb{N}}$ recursively by $F_0 := \emptyset$,

$$F_{n+1} := F_n \cup \bigcup_{\phi \in \{Z(0),\dots,Z(n)\}} (\phi, d^n(\phi)),$$

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

We set $F(n) := F_{n+1}(n)$. This is a total map $F : \mathbb{N} \to \mathcal{F}_0 \times \{\pm\}$, having the property $F(\mathbb{N}) = \cup_n \text{image}(F_n)$. F is computable by explicit verification, using the axioms of $\mathcal{S}$. □

Returning to the proof of the theorem. Let $K = K_{\mathcal{A}} : \mathcal{T} \to \mathcal{T}$ be as in Lemma 3.7. For $\phi \in \mathcal{F}_0$ let α_ϕ be as in (3.19). Define: $H : \mathcal{T} \times \mathbb{N} \to \mathcal{A} \times \{\pm\}$ by

$$H(T, n) := \begin{cases} (K_{\mathcal{A}}(T))'(k), & \text{if } n = 2k+1 \\ (\alpha_{pr_{\mathcal{F}_0} \circ F(k)}, pr_{\pm} \circ F(k)), & \text{if } n = 2k, \end{cases}$$

where $pr_{\mathcal{F}_0} : \mathcal{F}_0 \times \{\pm\} \to \mathcal{F}$, and $pr_{\pm} : \mathcal{F}_0 \times \{\pm\} \to \{\pm\}$ are the natural projections. H is computable directly by the axioms of $\mathcal{S}$. (Factor H as a composition of computable maps as previously.)

Let $Spec : \mathcal{T} \to \mathcal{T}$ be the computable map corresponding to H via Axiom 7. In particular, for each $T \in \mathcal{T}$, $Spec(T)$ encodes the map

$$T'_{spec} := H^T : \mathbb{N} \to \mathcal{A} \times \{\pm\}, \quad H^T(n) = H(T, n),$$

which by construction is speculative. Now, $Spec(T)$ satisfies the Properties 1, 2, 3 immediately by construction. It only remains to check Property 4.

Lemma 3.22. *Let $T \in \mathcal{T}_{\mathcal{A}}$, then T^s_{spec} is consistent unless for some $\phi \in G$*

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

Proof. Suppose that T^s_{spec} is inconsistent so that:

$$T^s \cup \{\alpha_{\phi_1}, \ldots, \alpha_{\phi_n}\} \vdash \alpha \wedge \neg\alpha$$

for some $\alpha \in \mathcal{A}$, and some $\phi_1, \ldots, \phi_n \in G$. Hence,

$$T^s \vdash \neg(\alpha_{\phi_1} \wedge \ldots \wedge \alpha_{\phi_n}).$$

But

$$\alpha_{\phi_1} \wedge \ldots \wedge \alpha_{\phi_n} \iff \forall m : \phi(m),$$

where ϕ is the formula with one free variable: $\phi(m) := \phi_1(m) \wedge \ldots \wedge \phi_n(m)$. Clearly $\phi \in G$, since $\phi_i \in G$, $i = 1, \ldots, n$. Hence, the conclusion follows. □

Suppose that T^s_{spec} inconsistent, then by the lemma above for some $\phi \in G$:

$$T^s \vdash \exists m : \neg\phi(m).$$

Since T^s is 1-consistent:

$$\exists m : T^s \vdash \neg\phi(m).$$

But ϕ is in G, and $T^s \vdash RA$ (recall Definition 1.4) so that $\forall m : T^s \vdash \phi(m)$ and so

$$\exists m : T^s \vdash (\neg\phi(m) \wedge \phi(m)).$$

So T^s is inconsistent, a contradiction, so T^s_{spec} is consistent. □

4. The stable halting problem

Let $\mathcal{D}_{\mathcal{T}} \subset \mathcal{T}$ be as in Definition 3.5 with respect to $B = \mathcal{T}$.

Definition 4.1. *For $T \in \mathcal{D}_{\mathcal{T}}$, T* **is T-decided**, *is a special case of Definition 3.12. Or more specifically, it means that the element $T \in \mathcal{T}$ is T'-decided. We also say that T is* not *T-decided, when $\neg$(T is T-decided) holds.*

Definition 4.2. *We call a map $D : \mathcal{T} \times \mathbb{N} \to \{\pm\}$* **Turing decision map**. *We say that such a D is* **stably sound on** *$T \in \mathcal{T}$ if*

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

We say that D is **stably sound** *if it is stably sound on all T. We say that D* **stably decides** *T if:*

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

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

The informal interpretation of the above is that each such D is understood as an operation with the properties:

- For each T, n $D(T, n) = +$ if and only if D "decides" at the moment n that the sentence $(T \in \mathcal{D}_{\mathcal{T}}) \wedge (T$ is not T-decided) is true.
- For each T, n $D(T, n) = -$ if and only if D cannot "decide" at the moment n the sentence $(T \in \mathcal{D}_{\mathcal{T}}) \wedge (T$ is not T-decided), or D "decides" that it is false.

In what follows for $T \in \mathcal{T}$, and D as above, $\Theta_{D,T}$ is shorthand for the sentence:

$$(T \in \mathcal{D}_{\mathcal{T}}) \wedge (T \text{ stably computes } D).$$

Lemma 4.3. *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 is D-decided then since D is stably sound on T, $T \in \mathcal{D}_{\mathcal{T}}$ and T is not T-decided, so if in addition $\Theta_{D,T}$ then T is not D-decided a contradiction. □

The following is the "stable" analogue of Turing's halting theorem.

Theorem 4.4. *There is no (stably) computable Turing decision map D that is stably sound and complete.*

Proof. Suppose otherwise, and let D be stably sound and complete. Then by the above lemma we obtain:

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

But it is immediate:

$$\forall T \in \mathcal{D}_{\mathcal{T}} : \Theta_{D,T} \vdash (\neg(T \text{ is } D\text{-decided}) \vdash \neg(T \text{ is } T\text{-decided})). \tag{4.6}$$

So combining (4.5), (4.6) above we obtain

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

But D is complete so $(T \in \mathcal{D}_{\mathcal{T}}) \wedge \neg(T \text{ is } T\text{-decided}) \implies T$ is D-decided and so:

$$\forall T \in \mathcal{D}_{\mathcal{T}} : \Theta_{D,T} \vdash (T \text{ is } D\text{-decided}).$$

Combining with (4.5) we get

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

which is what we wanted to prove. □

Theorem 4.7. *Suppose $F \subset \mathcal{A}$ is stably c.e. and sound theory, then there is a constructible (given a Turing machine stably computing F) true in the standard model of arithmetic sentence $\alpha(F)$, which F does not prove.*

The fact that such an $\alpha(F)$ exists, can be immediately deduced from Tarski undecidability of truth, as the set F must be definable in first order arithmetic by the condition that F is stably c.e. However, our sentence is constructible by very elementary means, starting with the definition of a Turing machine, and the basic form of this sentence will be used in the next section. The above is of course only a meta-theorem. This is in sharp contrast to the syntactic incompleteness theorems in the following section which are actual theorems of ZF.

Proof of Theorem 4.7. Suppose that F is stably c.e. and is sound. Let $(M, M_e) : \mathbb{N} \to \mathcal{A} \times \{\pm\}$ be a total Turing machine s.t. $F = M^s(N)$. Let $C(M_e)$ be as in Lemma 3.16. If we understand arithmetic as being embedded in set theory ZF in the standard way, then for each $T \in \mathcal{T}$ the sentence

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

is logically equivalent in ZF to a first order sentence in arithmetic, that we call $s(T)$. The corresponding translation map $s : \mathcal{T} \to \mathcal{A}$, $T \mapsto s(T)$ is taken to be computable. Indeed, this kind of translation already appears in the original work of Turing [1].

Define a Turing decision map D by

$$D(T, n) := (Dec_{\mathcal{A}}(C(M_e)))'(s(T), n)$$

for $Dec_{\mathcal{A}}$ as in (3.10) defined with respect to $B = \mathcal{A}$, and where C is as in Section 3. Then by construction, and by Axiom 4 in particular, D is computable by some Turing machine (D, D_e), we make this more explicit in the following Section 5.

Now D is stably sound by Lemma 3.13 and the assumption that F is sound. So by Lemma 4.3:

$$\neg(D_e \text{ is } D\text{-decided}).$$

In particular, $s(D_e)$ is not $Dec_{\mathcal{A}}(C(M_e))$-decided, and so $s(D_e)$ is not $C(M_e)$-stable (Lemma 3.13), i.e. $M \nvdash s(D_e)$.

On the other hand,

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

by definition. And so since $D_e \in \mathcal{D}_{\mathcal{T}}$ by construction, $s(D_e)$ is satisfied. Set $\alpha(M) := s(D_e)$ and we are done. □

5. Syntactic incompleteness for stably computable theories

Let $s : \mathcal{T} \to \mathcal{A}$, $T \mapsto s(T)$ be as in the previous section. Define

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

by $H(M, T, n) := (Dec_{\mathcal{A}}(C(Spec(M))))'(s(T), n)$. We can express H as the composition of the sequence of maps:

$$\mathcal{T} \times \mathcal{T} \times \mathbb{N} \xrightarrow{Dec_{\mathcal{A}} \circ C \circ Spec \times s \times id} \mathcal{T} \times \mathcal{A} \times \mathbb{N} \xrightarrow{id \times e_{\mathcal{A} \times \mathbb{N}}} \mathcal{T} \times \mathcal{U} \xrightarrow{U} \mathcal{U} \xrightarrow{e^{-1}_{\{\pm\}}} \{\pm\}, \tag{5.1}$$

where the last map is:

$$\Sigma \mapsto \begin{cases} \text{undefined}, & \text{if } \Sigma \notin \{\pm\}_e \\ e^{-1}_{\{\pm\}}(\Sigma), & \text{otherwise.} \end{cases}$$

So H is a composition of maps that are computable by the axioms of $\mathcal{S}$ and so H is computable. Hence, by Axiom 7 there is an associated computable map:

$$Tur : \mathcal{T} \to \mathcal{T}, \tag{5.2}$$

s.t. for each $M \in \mathcal{T}$, $Tur(M)$ encodes the map $D^M : \mathcal{T} \times \mathbb{N} \to \{\pm\}$, $D^M(T, n) = H(M, T, n)$.

In what follows, $(M, M_e) : \mathbb{N} \to \mathcal{A} \times \{\pm\}$ will be a fixed total Turing machine. We abbreviate D^{M_e} by D and $Tur(M_e)$ by D_e. As usual notation of the form $M \vdash \alpha$ means $M^s \vdash \alpha$.

Proposition 5.3. *For $(M, M_e), (D, D_e)$ as above:*

$$M^s \text{ is 1-consistent} \implies M \nvdash s(D_e).$$

$$M^s \text{ is 2-consistent} \implies M \nvdash \neg s(D_e).$$

Moreover, the sentence:

$$M^s \text{ is 1-consistent} \implies s(D_e)$$

is a theorem of PA under standard interpretation of all terms, (this will be further formalized in the course of the proof).

Proof. This proposition is meant to just be a theorem of set theory ZF, however we avoid complete set theoretic formalization, as is common. Arithmetic is interpreted in set theory the standard way, using the standard set $\mathbb{N}$ of natural numbers. So for example, for $M : \mathbb{N} \to \mathcal{A} \times \{\pm\}$ a sentence of the form $M \vdash \alpha$ is a priori interpreted as a sentence of ZF, however if M is a Turing machine this also can be interpreted as a sentence of PA, once Gödel encodings are invoked.

Set $N := (Spec(M_e))'$, in particular this is a speculative total Turing machine $\mathbb{N} \to \mathcal{A} \times \{\pm\}$. Set $s := s(D_e)$. Suppose that $M \vdash s$. Hence, $N \vdash s$ and so s is $C(Spec(M_e))$-stable, and so by Lemma 3.13 s is $Dec_{\mathcal{A}}(C(Spec(M_e)))$-decided, and so D_e is D-decided by definition. More explicitly, we deduce the sentence η_{M_e}:

$$\exists m \forall n \geq m : (Tur(M_e))'(Tur(M_e), m) = +.$$
$$\text{i.e. } \exists m \forall n \geq m : D(D_e, m) = +.$$

In other words:

$$(M \vdash s) \implies \eta_{M_e} \tag{5.4}$$

is a theorem of ZF.

If we translate η_{M_e} to an arithmetic sentence we just call $\eta = \eta(M_e)$, then η can be chosen to have the form:

$$\exists m \forall n : \gamma(m,n),$$

where $\gamma(m,n)$ is RA-decidable. The sentence $s = s(D_e)$ is assumed to be of the form $\beta(M_e) \wedge \neg\eta(M_e)$, where $\beta(M_e)$ is the arithmetic sentence equivalent in ZF to $(D_e = Tur(M_e) \in \mathcal{D}_T)$. Clearly, the translation maps $\mathcal{T} \to \mathcal{A}$, $T \mapsto \beta(T)$, $T \mapsto \eta(T)$ can be taken to be computable, and such that applying Lemma 3.11 (interpreted as a Theorem of PA): we get

$$PA \vdash \forall T \in \mathcal{T} : \beta(T). \tag{5.5}$$

And so

$$PA \vdash (\eta(M_e) \implies \neg s(D_e)). \tag{5.6}$$

Moreover, ZF proves:

$$\begin{aligned} \eta_{M_e} &\implies \exists m \forall n : RA \vdash \gamma(m,n), \quad \text{trivially} \\ &\implies \exists m : N \vdash \forall n : \gamma(m,n), \quad \text{since } N \text{ is speculative} \\ &\implies N \vdash \eta, \\ &\implies N \vdash \neg s, \quad \text{by (5.6) and since } N^s \supset PA. \end{aligned}$$

And so combining with (5.4), (5.6) ZF proves:

$$(M \vdash s) \implies (N \vdash s) \wedge (N \vdash \neg s).$$

Since by Theorem 3.20

$$M^s \text{ is 1-consistent} \implies N^s \text{ is consistent},$$

it follows:

$$ZF \vdash (M^s \text{ is 1-consistent} \implies M \nvdash s \quad (\text{moreover } N \nvdash s)). \tag{5.7}$$

Now suppose

$$(M^s \text{ is 2-consistent}) \wedge (M \vdash \neg s).$$

Since we have (5.5), and since $M \vdash PA$ it follows that $M \vdash \eta$.

Now,

$$\begin{aligned} M \vdash \eta &\iff M \vdash \exists m \forall n : \gamma(m,n), \\ &\implies \neg(\forall m : M \vdash \neg\phi(m)) \quad \text{by 2-consistency}, \end{aligned}$$

where

$$\phi(m) = \forall n : \gamma(m,n).$$

So we deduce

$$\exists m : M \vdash \phi(m).$$

Furthermore,

$$\begin{aligned} \exists m : M \vdash \phi(m) &\implies \exists m \forall n : M \vdash \gamma(m,n), \quad M^s \text{ is consistent} \\ &\implies \exists m \forall n : RA \vdash \gamma(m,n), \quad M^s \text{ is consistent, } M^s \supset RA \text{ and } \gamma(m,n) \text{ is } RA\text{-decidable}. \end{aligned}$$

In other words, ZF proves:

$$(M^s \text{ is 2-consistent} \wedge (M \vdash \neg s)) \implies \eta.$$

And ZF proves:

$$\eta \implies N \vdash s,$$

by constructions. So ZF proves:

$$\begin{aligned}(M^s \text{ is 2-consistent} \wedge (M \vdash \neg s)) &\implies N \vdash s \\ &\implies N^s \text{ is inconsistent} \\ &\implies M^s \text{ is not 1-consistent, by Theorem 3.20} \\ &\implies M^s \text{ is not 2-consistent.}\end{aligned}$$

And so ZF proves:

$$M^s \text{ is 2-consistent} \implies M \nvdash \neg s.$$

Now for the last part of the proposition. We essentially just further formalize (5.7) and its consequences in PA. In what follows by equivalence of sentences we mean equivalence in ZF. The correspondence of sentences under equivalence is the standard kind of correspondence assigning predicates involving Turing machines predicates in PA. The basic form of such correspondences is already constructed by Turing [1], so that we will not elaborate. In particular, the correspondences are computable, which just means that the corresponding map $\mathcal{T} \to \mathcal{A}$ is computable.

Definition 5.8. *We say that $T \in \mathcal{T}$ is* **stably 1-consistent** *if $T \in \mathcal{T}_{\mathcal{A}}$, T' is total and T^s is 1-consistent, (Notation 2). The sentence "T^s is 1-consistent", can specifically be taken to be the arithmetic sentence $Con_{\sigma,1}$ for σ the natural Σ_2 definition of T^s and $Con_{\sigma,1}$ the consistency sentence as in* [7, Section 5].

Then the sentence:

$$T \text{ is stably 1-consistent}$$

is equivalent to an arithmetic sentence we denote:

$$1 - con^s(T).$$

The sentence $Spec(T) \nvdash s(Tur(T))$ is equivalent to an arithmetic sentence we call:

$$\omega(T).$$

By the proof of the first part of the proposition, that is by (5.7),

$$ZF \vdash \forall T \in \mathcal{T} : (1 - con^s(T) \implies \omega(T)). \tag{5.9}$$

But we also have:

$$PA \vdash \forall T \in \mathcal{T} : (1 - con^s(T) \implies \omega(T)), \tag{5.10}$$

since the first part of the proposition can be formalized in PA, in fact the only interesting theorems we used are Lemma 3.13, and Theorem 3.20 which are obviously theorems of PA. By Lemma 3.13 and the construction of H:

$$PA \vdash \forall T \in \mathcal{T} : \omega(T) \iff \neg\eta(T).$$

So:

$$PA \vdash \forall T \in \mathcal{T} : (\beta(T) \wedge \omega(T) \iff s(Tur(T))),$$

Combining with (5.5) and with (5.10) we get:

$$PA \vdash \forall T \in \mathcal{T} : (1 - con^s(T) \implies s(Tur(T))).$$

So if we formally interpret the sentence "M^s is 1-consistent" as the arithmetic sentence $1 - con^s(M_e)$, then this formalizes and proves the second part of the proposition. □

Proof of Theorems 1.5, 1.6. Let F be as in the hypothesis. As F is stably c.e. we may find a Turing machine (M, M_e) such that $F = M^s$. We may in addition assume that M is total, by classical theory. Let (D, D_e) be as in the proposition above, and set $\alpha(F) := s(D_e)$. Then Theorem 1.5 follows by part one of the proposition above. Theorem 1.6 follows by part two of the proposition above. □

Proof of Theorem 1.9. Let $F, F_{\mathcal{A}}$ be as in the hypothesis. By (5.7), and by (5.5)

$$ZF \vdash \forall T \in \mathcal{T}_{\mathcal{A}} : (T \text{ is total and } T^s \text{ is 1-consistent }) \implies s(Tur(T)). \tag{5.11}$$

Suppose that

$$F \vdash F_{\mathcal{A}} \text{ is 1-consistent},$$

and

$$\exists T : F \vdash T \text{ stably computes } F_{\mathcal{A}}.$$

Let's call this the main assumption. Then also for some $T' \in \mathcal{T}_{\mathcal{A}}$

$$F \vdash T' \text{ is total and stably computes } F_{\mathcal{A}},$$

where T' is such that:

$$ZF \vdash (T' \text{ is total}) \wedge ((T' \text{ stably computes } F_{\mathcal{A}}) \iff (T \text{ stably computes } F_{\mathcal{A}})). \tag{5.12}$$

Since $F \vdash ZF$ by assumption, by (5.11) $F \vdash s(Tur(T'))$. More specifically, $ZF \vdash (F_{\mathcal{A}} \vdash s(Tur(T')))$. But also

$$ZF \vdash ((F_{\mathcal{A}} \vdash s(Tur(T'))) \implies \neg(F_{\mathcal{A}} \text{ is 1-consistent}) \vee \neg(T' \text{ stably computes } F_{\mathcal{A}})).$$

Since $F \vdash ZF$, we conclude that

$$F \vdash (\neg(F_{\mathcal{A}} \text{ is 1-consistent}) \vee \neg(T' \text{ stably computes } F_{\mathcal{A}})).$$

And so by (5.12):

$$F \vdash (\neg(F_{\mathcal{A}} \text{ is 1-consistent}) \vee \neg(T \text{ stably computes } F_{\mathcal{A}})).$$

Then by the main assumption F is inconsistent.

□

Appendix A. Stable computability and physics - Gödel's disjunction and Penrose

In this appendix we give some additional background, to give "real world" motivation to the idea of stable computability. In the language of the paper, this discussion is relatively new. We try to be very concise.

We may say that a physical process is *absolutely not Turing computable*, if it is not Turing computable in any "sufficiently physically accurate" mathematical model. 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 both of the processes above may become computable, possibly if the nature of the universe is ultimately discreet.

The question posed by Turing [2], but also by Gödel [6, 310] and more recently and much more expansively by Penrose [10], [11], [12] is:

Question 1. Are there absolutely not Turing computable physical processes? And moreover, are brain processes absolutely not Turing computable?

A.0.1. *Gödel's disjunction.* Gödel argued for a 'yes' answer to Question 1, see [6, 310], relating the question to existence of absolutely unsolvable Diophantine problems, see also Feferman [5], and Koellner [8], [9] for a discussion.

Let $F \subset \mathcal{A}$ be the mathematical output of a certain idealized mathematician S (where for the moment we leave the latter undefined). Let A be the sentence asserting that F, as a theory, can decide any Diophantine problem. Essentially, Gödel argues for a disjunction, the following cannot hold simultaneously:

$$F \text{ is computably enumerable, } F \text{ is consistent and } A,$$

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.

Claim 1. It is impossible to meaningfully formalize Gödel's disjunction without strengthening the incompleteness theorems.

Already Feferman asks in [5] what is the meaning of 'idealized' above? In the context of the present paper, let's model the mathematical output of S as a partial function of time $\mathbb{N}$, so $S : \mathbb{N} \to \mathcal{A} \times \{\pm\}$. We might then first propose that 'idealized' just means stabilized (as in the Introduction, and Section 3). But there are Turing machines T whose stabilization T^s is not computably enumerable, as shown in Example 3.3. In that case, the above Gödel disjunction becomes meaningless because passing to the idealization may introduce non-computability where there was none before.

If we give up computability of the ideal limit, but still want to salvage the disjunction of Gödel then we must suitably extend the incompleteness theorems.

A.0.2. *Formalizing Gödel's disjunction.* We must first propose a suitable idealization to which we may apply the incompleteness theorems of this paper. One immediate possibility is to simply substitute humanity (evolving in time), in place of a subject. This requires its own implicit idealizations that we ignore, but fits well with our expectation that overall human knowledge converges on truth. Thus, we suppose that associated to humanity there is a function:

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

determined, for example, by majority consensus. Majority consensus is not as restrictive as it sounds. For instance, if at a moment n there is a computer verified proof of the Riemann hypothesis α in ZFC, then irrespectively of the complexity of the proof, we can expect majority consensus for α, provided validity of set theory still has consensus. And then $H(n) = (\alpha, +)$.

In what follows, $\mathcal{H} = H^s$, for H as outlined above. So that informally $\mathcal{H}$ is the infinite time limit of the mathematical output of humanity in the language of arithmetic. There is clearly some freedom for how the underlying map H is interpreted and constructed, however $\mathcal{H}$ itself is at least morally unambiguous.

Instead of trying to totally specify H, we will instead assume the following axiom:

Axiom A.1. *If the physical processes underlying the output of H are computable, this may include computability of certain stochastic variables as probability distribution valued maps - then the set $\mathcal{H}$ is stably computably enumerable.*

Applying Corollary 1.8 we then get the following.

Theorem A.2. *(Conditional on the axiom above.) Assume the axiom above. Suppose that $\mathcal{H} \vdash PA$ and $\mathcal{H}$ is 1-consistent. Then either there are absolutely non Turing computable processes in nature or there exists a true in the standard model* constructive *statement of arithmetic $\alpha(\mathcal{H})$ that $\mathcal{H}$ cannot prove (and cannot disprove if $\mathcal{H}$ is in addition 2-consistent). (By constructive we mean provided a specification of a Turing machine stably computing $\mathcal{H}$.)*

It would be good to compare this theorem with Deutsch [4], 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 in our case.

A.0.3. *Turing.* Finally, let us return to Turing himself. He suggested in [2] that abandoning hope of consistency is the obvious way to circumvent the implications of Gödel incompleteness theorem for computability of intelligence. In view of the Theorem A.2 just above, it appears that this position is untenable. Humans may not be consistent, but it is implausible that the stabilization $\mathcal{H}$ is not sound, since as we say human knowledge appears to converge on truth, and it is rather inconceivable that $\mathcal{H}$ is not 1-consistent. Then we cannot escape incompleteness by the above. So if one insists on computability, then the only plausible 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.

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

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

[6] K. Gödel, *Collected Works III (ed. S. Feferman)*, New York: Oxford University Press, 1995.

[7] 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.

[8] 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.

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

[10] R. Penrose, *Shadows of the mind*, 1994.

[11] ———, *Beyond the shadow of a doubt*, Psyche, (1996).

[12] ———, *Road to Reality*, 2004.

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

[14] R. I. Soare, *Turing computability. Theory and applications*, Berlin: Springer, 2016.

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