Direct link to author's version

INCOMPLETENESS THEOREMS VIA TURING CATEGORY

YASHA SAVELYEV

Abstract. We give a set theoretic reframing of Gödel's first and second incompleteness theorems that applies even to some undefinable theories of arithmetic. The usual Hilbert–Bernays provability conditions and the "diagonal lemma" are absent, replaced by a more direct diagonalization argument, from first principles, based in set theory/category theory and in a sense analogous to Cantor's original argument. To this end, we categorify the theory Gödel encodings, which might be of independent interest. In our setup, the Gödel sentence is computable explicitly by construction even for Σ_2^0 theories (likely extending to Σ_n^0). In an appendix, we study the relationship of our reframed second incompleteness theorem with arguments of Penrose.

1. Introduction

Ordinarily, Gödel incompleteness theorems are formulated for definable arithmetic theories. In fact, the original incompleteness theorems applied to computably enumerable theories of arithmetic, but the latter can be generalized in some ways to Σ_n theories, see for instance [7]. One of the goals here is to extend first and second incompleteness theorems to certain undefinable theories of arithmetic. Another goal is to rework the incompleteness theorems from a more set theoretic/categorical point of view, utilizing a certain category of Gödel encodings. The extension to some undefinable theories is then one proof of concept, for this recasting of the incompleteness theorems.

We will work from first principles, and within the meta-theory ZFC, that is all theorems are theorems of ZFC (and mostly just ZF). Our setup will be partly language agnostic, we work with a general $\mathcal{L}$-theory, interpreting arithmetic but our notion of interpretation is very weak. Partly for this reason various standard tools like Hilbert-Bernays provability conditions and the "diagonal lemma" will not appear, as translating them to our context is difficult. Instead, the diagonalization argument that we use is more direct, in a sense elementary, and more analogous to the set theoretic diagonalization argument of Cantor, as well as to Turing's original work on the halting problem. To this end we also categorify the theory of Gödel encodings, obtaining what we call the Turing category. This might be of independent interest.

If we simplify some assumptions [1], our argument drastically simplifies, and may be of interest purely as another, from first principles of Turing machines approach to the classical incompleteness theorems.

The primary motivation and many of the choices we make, most notably the choice to work with stably c.e. theories (see following section), are based around the problem of formalizing a certain version of an argument of Roger Penrose on potential non-computability in physics. The reader may see Appendix A, however many terms there are defined throughout the paper.

1.1. Definition of stably c.e. theories and statements of results.

Let $\mathcal{L}$ be a first order language, in particular a formal language in a countable alphabet including the symbols of first order logic, with syntax of first order logic, such that elements of $\mathcal{L}$ are sentences (in the logical sense).

We now quickly introduce stable c.e. theories. In what follows, a map is always a partial map, unless it is specified to be total. Suppose we are given a map

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

[1] For example restrict to c.e. theories of arithmetic as in the original incompleteness theorems.

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

Definition 1.1. *We say that $\alpha \in \mathcal{L}$* **is M-stable** *if there is an m with $M(m) = (\alpha, +)$ s.t. there is no $n > m$ with $M(n) = (\alpha, -)$. Let $M^s \subset \mathcal{L}$ denote the set of M-stable α, called* **the stabilization of** *M.*

Remark 1.2. *For an informal motivation of how such an M may appear in practice consider the following. With $\mathbb{N}$ playing the role of time, M might be a mathematician producing sentences of arithmetic that it believes to be true, at each moment $n \in \mathbb{N}$. But M is also allowed to correct itself in the following sense.*

- *$M(n) = (\alpha, +)$, only if at the moment n M decides that α is true.*
- *$M(m) = (\alpha, -)$, only if at the moment m, M no longer asserts that α is true, either because at this moment M is no longer able to decide α, or because it has decided it to be false.*

Definition 1.3. *A subset $S \subset \mathcal{L}$, is called* **stably computably enumerable** *or stably c.e., if there is a computable map (see Definition 2.3) $T : \mathbb{N} \to \mathcal{L} \times \{\pm\}$ so that $S = T^s$. In this case we also say that T* **stably enumerates** *S. We say that $T \in \mathcal{T}$* **stably computes** *$M : \mathbb{N} \to \mathcal{L} \times \{\pm\}$, if it computes some $N : \mathbb{N} \to \mathcal{L} \times \{\pm\}$, s.t. $M^s = N^s$.*

It is fairly immediate that a stably c.e. S is $\Sigma_2 = \Sigma_2^0$ definable. The converse is also true, every Σ_2 definable set $S \subset \mathcal{L}$ is stably c.e.. To prove this we may build on Example 3.3, to construct an oracle and then use the theorems of Post, (see [13]), relating the arithmetic hierarchy with the theory of Turing degrees. We omit the details as this will not be essentially used, and is a well understood idea.

Let $\mathcal{A}$ denote the first order language of arithmetic with non-logical symbols $\{0, +, \times, s, <\}$). In the following we need a notion of a n-translation from the language of arithmetic to another abstract language, as well as the notion of n-consistency and strong consistency. The preliminaries for this are given in Section 2.2.

Note that because our notion of a translation is weak, we cannot reduce the following to the second incompleteness theorems for definable arithmetic theories. This is because the corresponding arithmetic theory, that is the theory $F_{i,\mathcal{A}}$ in the notation of Section 2.2, will generally not be arithmetically definable.

Theorem 1.4. *Let F be a theory in any language $\mathcal{L}$, such that $F \vdash^i ZFC$ for a 2-translation i. Then if F is strongly consistent:*

$$(F \nvdash^i F \text{ is 1-consistent}) \vee (F \nvdash^i F \text{ is stably c.e., i.e. is } \Sigma_2). \tag{1.5}$$

The above is based on the following results. The next theorem in particular tells us that the Gödel sentence is "computable", even in the setup of abstract $\mathcal{L}$-theories above.

Theorem 1.6. *Let $\mathcal{L}$ be any fixed first order language, and let $i : \mathcal{A} \to \mathcal{L}$ be some 2-translation. Then there is a total computable map*

$$\mathcal{G} : \mathcal{T} \to \mathcal{A},$$

(depending only on i) where the domain $\mathcal{T}$ is the set of Turing machines $\mathbb{N} \to \mathbb{N}$, and where $\mathcal{G}$ satisfies the following. Suppose that T stably computes *$M : \mathbb{N} \to \mathcal{L} \times \{\pm\}$. Let $F = M^s$, then we have:*

(1) *$F \nvdash^i \mathcal{G}(T)$ if F is 1-consistent.*

(2) *$F \nvdash^i \neg\mathcal{G}(T)$ if F is 2-consistent.*

(3)

$$(F \text{ is 1-consistent}) \implies \mathcal{G}(T). \tag{1.7}$$

Furthermore, the Turing machine computing $\mathcal{G}$, can itself be given constructively.

As a corollary we get a more basic form of Theorem 1.4:

Corollary 1.8. *For F a theory in any language $\mathcal{L}$ such that $F \vdash^i ZFC$, where i is a 2-translation:*

$$\forall T \in \mathcal{T}\ ((F \text{ is 1-consistent}) \wedge (F \vdash^i T \text{ stably enumerates } F) \implies (F \nvdash^i F \text{ is 1-consistent})).$$

1.2. **Generalizations to Σ_n.** There are natural candidates for how to generalize the above. We may replace $M : \mathbb{N} \to \mathcal{L} \times \{\pm\}$ by $M : \mathbb{N}^n \to \mathcal{L} \times \{\pm\}$, using this we can define a notion of n-stable computability, specializing to stable computability for $n = 1$. The above theorems should generalize to this setting of n-stable computability. In terms of arithmetic complexity this should be exactly the class Σ^0_{n+1}. We leave this for future developments.

2. Some preliminaries

2.1. **Abstractly encoded sets and the Turing category.** The material of this section will be used in the main argument. The approach here is in essence the *standard* approach, but usually encoding systems are implicit and fixed, as ordinarily one is working with some concrete sets and concrete encodings. We need to work with abstract sets (in particular all abstract first order languages), and we need to formalize the properties of encodings in such a way that showing maps are computable can be done axiomatically, as the maps we construct are fairly complex. For this, it is very natural to use the language of category theory.

Definition 2.1. *An* **encoding map** *of a set A is an injective total set map $e : A \to \mathbb{N}$, such that the set $A_e = e(A)$ is computable (recursive).*

Here, as is standard, a set $S \subset \mathbb{N}$ is called *computable* if both S and its complement are computably enumerable, with S called *computably enumerable* if there is a computable partial function $\mathbb{N} \to \mathbb{N}$ with range S.

We extend the collection of encodings to a structure of a category.

Definition 2.2. *We denote by $\mathcal{T}$ the set of all Turing machines $T : \mathbb{N} \to \mathbb{N}$. We write $*T(n)$ for the computation sequence of the Turing machine T with input n. As usual, for $T \in \mathcal{T}$, T also denotes the underlying partial function with $T(n) = m$ if $*T(n)$ halts with output m, and undefined otherwise.*

In what follows, a map is a partial map, unless we specify that it is total. Define a large (the set of objects is a proper class) arrow type category $\mathcal{S}$ whose objects are encoding maps $e_A : A \to \mathbb{N}$. Explicitly, objects $\operatorname{obj}\mathcal{S}$ of $\mathcal{S}$ consists of pairs (A, e_A) where A is a set, and $e_A : A \to \mathbb{N}$ an encoding map.

We now describe the morphisms of $\mathcal{S}$.

Definition 2.3. *For (N, e_N), (M, e_M) in $\operatorname{obj}\mathcal{S}$, a morphism is a map f s.t. there is a commutative diagram:*

$$\begin{array}{ccc} N & \xrightarrow{f} & M \\ \downarrow{\scriptstyle e_N} & & \downarrow{\scriptstyle e_M} \\ \mathbb{N} & \xrightarrow{T} & \mathbb{N}, \end{array}$$

for some $T \in \mathcal{T}$.

To simplify notation, we may omit specifying the encoding map for a given object of $\mathcal{S}$, keeping track of it implicitly. With this in mind, for $N, M \in \mathcal{S}$ we say that T ***computes*** a map $f : N \to M$ if T fits into a commutative diagram as above. We say that $f : N \to M$ is ***computable*** if there exists a $T \in \mathcal{T}$ which computes f.

So in the terms above, the set $hom_{\mathcal{S}}((N, e_N), (M, e_M))$ is the set of computable maps $f : N \to M$. In what follows, if not specified, the encoding of $\mathbb{N}$ is the $id : \mathbb{N} \to \mathbb{N}$.

Proposition 2.4. *The category $\mathcal{S}$, called the* **Turing category***, satisfies the following:*

(1) *The category $\mathcal{S}$ has finite products. (In fact it has all finite limits and colimits, but we will not need this.) This entails the following. If $(A, e_A), (B, e_B) \in \mathcal{S}$ then there is a distinguished encoding map $e_{A\times B} : A \times B \to \mathbb{N}$, called the* **product encoding map** *s.t.:*

   (a) *The projection maps $pr^A : A \times B \to A$, $pr^B : A \times B \to B$ are computable. Similarly for pr^B.*

   (b) *If $f : A \to B$ is computable, and $g : A \to C$ is computable then $A \to B \times C$, $a \mapsto (f(a), g(a))$ is computable.*

   (c) *If $f : A \to B$, $g : C \to D$ are computable then the map $A\times B \to C\times D$, $(a, b) \mapsto (f(a), g(b))$ is computable.*

   From now on, the encoding map of $A \times B$ is taken to be the product encoding map (with respect to some possibly implicit encoding maps of A, B).

(2) *The set $\mathcal{T}$ has a distinguished encoding $e_{\mathcal{T}}$ s.t. the following holds. Define*

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

$$U(T, \Sigma) := \begin{cases} T(\Sigma), & \text{if } *T(\Sigma) \text{ halts} \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

   Then U is computable. (The Turing machine computing U may be called the "universal Turing machine".) From now $\mathcal{T}$ as an object $\mathcal{S}$ is taken with respect to this distinguished encoding.

(3) *Let $A, B, C \in \mathcal{S}$, and suppose that $f : A \times B \to C$ is computable. Let $f^a : B \to C$ be the map $f^a(b) = f(a, b)$. Then there is a computable map*

$$s : A \to \mathcal{T}$$

   so that for each a $s(a)$ computes f^a.

(4) *For a set A let*

$$L(A) := \bigcup_{n\in\mathbb{N}} Maps(\{0, \dots, n\}, A),$$

   where $Maps(\{0, \dots, n\}, A)$ denotes the set of total maps. If $(A, e_A) \in \mathcal{S}$ then there is a distinguished encoding $e_{L(A)}$ of $L(A)$ s.t.:

   (a) *The length function*

$$\text{length} : L(A) \to \mathbb{N},$$

   is computable, where for $l \in L(\mathcal{A})$, $l : \{0, \dots n\} \to A$, $\text{length}(l) = n$.

   (b) *Define*

$$P : L(A) \times \mathbb{N} \to A,$$

$$P(l, i) := \begin{cases} l(i), & \text{if } 0 \le i \le \text{length}(l) \\ \text{undefined}, & \text{for } i > \text{length}(l). \end{cases}$$

   Then P is computable.

   (c) *For $A, B \in \mathcal{S}$ and $f : A \to L(B)$ a partial map, suppose that:*

- *The partial map* $A \times \mathbb{N} \to B$, $(a, n) \mapsto P(f(a), n)$ *is computable.*
- *The partial map* $A \to \mathbb{N}$, $a \mapsto \mathrm{length}(f(a))$ *is computable.*

Then f *is computable. From now, given* $A \in \mathcal{S}$ *the encoding of* $L(A)$ *is assumed to be such a distinguished encoding, called the* **list encoding map**.

(5) *Let* $\mathcal{L}$ *be a first order language, then there is a distinguished encoding of* $\mathcal{L}$ *with the following property. There is a total computable map:*

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

s.t. for each $l \in L(\mathcal{L})$, $\Phi(\{l\} \times \mathbb{N})$ *is the deductive closure of the theory* $F_l = \mathrm{image}\, l$. *From now such an* $\mathcal{L}$ *is assumed to have such a distinguished encoding.*

Lemma 2.5. *If* $f : A \to B$ *is computable then the map* $L(f) : L(A) \to L(B)$,

$$l \mapsto \begin{cases} i \mapsto f(l(i)), & \text{if } f(l(i)) \text{ is defined for all } 0 \leq i \leq \mathrm{length}(l) \\ undefined, & \text{otherwise,} \end{cases}$$

is computable. Also, the map $LU : \mathcal{T} \times L(\mathcal{U}) \to L(\mathcal{U})$,

$$l \mapsto \begin{cases} i \mapsto U(T, (l(i))), & \text{if } U(T, (l(i))) \text{ is defined for all } 0 \leq i \leq \mathrm{length}(l) \\ undefined, & \text{otherwise} \end{cases}$$

is computable.

Proof. This is just a straightforward application of the proposition and part 4 in particular. We leave the details as an exercise. □

Proof of Proposition 2.4. As this is just an elaboration on classical theory we only sketch the proof. To prove the first part, given $(A, e_A), (B, e_B)$ the encoding map $e_{A \times B} : A \times B \to \mathbb{N}$ can be taken to be the map $(a, b) \mapsto 2^{e_A(a)} \cdot 3^{e_B(b)}$. The needed properties readily follow.

The second part is just the classical story of the universal Turing machine.

The third part corresponds to the "s-m-n theorem" Soare [13, Theorem 1.5.5], which works as follows. Given a classical 2-input Turing machine

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

there is a Turing machine $s_T : \mathbb{N} \to \mathbb{N}$ s.t. for each m $s_T(m)$ is the Turing-Gödel encoding natural, of a Turing machine computing the map f^m: $n \mapsto T(m, n)$.

For the fourth part we can just explicitly construct the needed encoding, by setting $e_{L(A)}$ to be the map

$$l \mapsto 2^{e_A(l(0))} \cdot \ldots \cdot p_n^{e_A(l(n))},$$

where $l : \{0, \ldots, n\} \to A$ and p_n is the n'th prime.

The last part follows by basic theory of first order logic and the previous parts. □

Remark 2.6. *The above properties suffice for our purposes. As mentioned* $\mathcal{S}$ *in fact has all finite limits and colimits, this is proved analogously. For example the sum (coproduct) of* (A, e_A) *and* (B, e_B), *can be given by* $(A \sqcup B, e_{A \sqcup B})$ *where*

$$e_{A \sqcup B}(x) = \begin{cases} 2^{e_A(x)}, & \text{if } x \in A \\ 3^{e_B(x)}, & \text{if } x \in B. \end{cases}$$

$\mathcal{S}$ having finite products and sums is in part what it makes it possible to have a computer programming language with algebraic data types, e.g. Haskell. Haskell also has more general, finite "colimit, limit" data types.

2.2. **Some preliminaries on first order theories.** Let $\mathcal{L}$ be a first order language. An $\mathcal{L}$-theory F is a subset $F \subset \mathcal{L}$. F will be called *deductively complete* if it is closed under inference, that is if $F \vdash \alpha$ then $\alpha \in F$. Denote by $\overline{F}$ the deductive closure of F.

$$\overline{F} = \cap\{H \subset \mathcal{L} \,|\, H \text{ is deductively complete and } F \subset H\}.$$

We will need a notion of one first order theory interpreting another first order theory, possibly in a different language.

Definition 2.7. *Given first order languages $\mathcal{L}, \mathcal{L}'$, a* **translation** *is a total set embedding $i : \mathcal{L}' \to \mathcal{L}$ such that*

(1) *i preserves the logical operators $\wedge, \vee, \neg$. (For example, $i(\neg\alpha \wedge \beta) = \neg i(\alpha) \wedge i(\beta)$).*

(2)
$$(\forall \alpha \in \mathcal{L}' \; \forall S \subset \mathcal{L}') \; S \vdash \alpha \implies i(S) \vdash i(\alpha).$$

Here is one well understood example.

Example 2.8. If $\mathcal{Z}$ denotes the first order language of set theory and $\mathcal{A}$ the first order language of arithmetic as in the introduction, then there is a translation $i_{\mathcal{A},\mathcal{Z}} : \mathcal{A} \to \mathcal{Z}$. This map assigns to numerals in the language of arithmetic the Von Neumann naturals, e.g. we assign $\emptyset$ to 0. In this example, assuming standard encodings of $\mathcal{A}, \mathcal{Z}$, i is computable.

If i as above is computable we call i a ***computable translation map***. The following, weakens this notion.

Definition 2.9. *For a translation $i : \mathcal{A} \to \mathcal{L}$, if the restriction of i to the subset of Σ^0_n formulas is computable we say that i is a n-***translation***. Likewise, for a translation $j : \mathcal{Z} \to \mathcal{L}$ we say it is a n-translation, if the induced translation $j \circ i_{\mathcal{A},\mathcal{Z}} : \mathcal{A} \to \mathcal{L}$, where $i_{\mathcal{A},\mathcal{Z}}$ is as above, is a n-translation.*

Definition 2.10. *Given a translation $i : \mathcal{L}' \to \mathcal{L}$ and an $\mathcal{L}$-theory F, we set*

$$F_{i,\mathcal{L}'} = i^{-1}(\overline{F}) \subset \mathcal{L}'.$$

To paraphrase, this is set of $\mathcal{L}'$-sentences proved by F under the given translation.

The following is immediate from the definitions.

Lemma 2.11. *For $\mathcal{L}, \mathcal{L}', i, F$ as above $F_{i,\mathcal{L}'}$ is a deductively closed theory.*

Definition 2.12. *For a given translation $i : \mathcal{L}' \to \mathcal{L}$, given an $\mathcal{L}$-theory F and a sentence $\alpha \in \mathcal{L}'$, we write $F \vdash^i \alpha$ if*

$$F_{i,\mathcal{L}'} \vdash \alpha.$$

Likewise, we write $F \vdash^i F'$ if

$$F_{i,\mathcal{L}'} \vdash F'.$$

Whenever, there exists an i s.t. $F \vdash^i F'$, we say that F **interprets** *F'.*

The following is also immediate from definitions.

Lemma 2.13. *Given first order languages $\mathcal{L}_0, \mathcal{L}_1, \mathcal{L}_2$ and an $\mathcal{L}_2$-theory F_2. If F_2 interprets F_1 and F_1 interprets F_0 then F_2 interprets F_0.*

Let $\mathcal{F}_0$ denote the set of Σ^0_0 formulas of arithmetic $\mathcal{A}$ with one free variable.

Definition 2.14 (cf. [4]). *Given a first order theory F in any language $\mathcal{L}$, we say that F* **1-consistent relative to the translation** *i if:*

(1) *$F \vdash^i Q$ as in Definition 2.12.*

(2) *For any formula $\phi \in \mathcal{F}_0$ the following holds:*

$$F \vdash^i \exists m\, \phi(m)) \implies (\exists m\, F \nvdash^i \neg\phi(m)).$$

We say that F is **2-consistent relative to the translation** *i if the same holds for Π^0_1 formulas ϕ with one free variable, more specifically formulas $\phi = \forall n\, g(m,n)$, with g Σ^0_0.*

From now on n-consistency is always with respect to some implicit translation i, that should be clear from the context, and so may not be denoted.

Definition 2.15. *Suppose we are given a theory F in some language $\mathcal{L}$, such that $F \vdash^i ZFC$. Then we say that it is* **strongly consistent** *if there is a 'standard model' M for $F_{i,\mathcal{Z}}$. More specifically, we suppose that M is a substructure of V_κ for V_κ some stage in the Von Neumann hierarchy.*

3. Stable computability and decision maps

In this section, general sets, often denoted as B, are intended to be objects of $\mathcal{S}$ with an implicit encoding map (sometimes made explicit). All maps are partial maps, unless specified otherwise. The set $\{\pm\}$ is always understood to be with the fixed encoding map $e_{\{\pm\}}(-) = 0$, $e_{\{\pm\}}(+) = 1$.

Definition 3.1. *Given 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 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. When M is computable, that is furnishes a morphism in $\mathcal{S}$, we say that $S \subset B$ is* **stably c.e.** *if $S = M^s$. We say that $T \in \mathcal{T}$* **stably computes** *$M : \mathbb{N} \to B \times \{\pm\}$, if it computes $N : \mathbb{N} \to B \times \{\pm\}$, s.t. $M^s = N^s$.*

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, with a distinguished encoding map whose properties will be specified shortly. 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.

Fix a distinguished encodings of $Pol, \mathbb{Z}$ so that the map

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

is computable. Let

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

be any total bijective computable maps.

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

$$l_1 : \{0, \dots, n\} \to B, \quad l_2 : \{0, \dots, m\} \to B$$

are two lists then $l_1 \cup l_2$ is defined by:

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

If $B \in \mathcal{S}$, it is easy to see that

$$\cup : L(B) \times L(B) \to L(B), \quad (l, l') \mapsto l \cup l'$$

is computable, given that we are using the list encoding map for $L(A)$, as in part 4c of Proposition 2.4.

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.

Define $A(n) := A_{n+1}(n)$. 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 that with this definition $A(\mathbb{N}) = \cup_{n \in \mathbb{N}} \text{image}(A_n)$.

Since E is computable, utilizing the recursive program above and Proposition 2.4, it can be readily verified that the map A is computable. 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 in the incompleteness theorems, and we now develop some of their theory.

Definition 3.5. *Let $B \in \mathcal{S}$, define $\mathcal{D}_B$ to be the set of $T \in \mathcal{T}$ s.t. exists $T' : B \times \mathbb{N} \to \{\pm\}$, and a commutative diagram:*

$$\begin{tikzcd} B \times \mathbb{N} \arrow[r, "T'"] \arrow[d, "e_{B \times \mathbb{N}}"] & \{\pm\} \arrow[d, "e_{\{\pm\}}"] \\ \mathbb{N} \arrow[r, "T"] & \mathbb{N}. \end{tikzcd}$$

More concretely, this is the set of T s.t.:

$$(\forall n \in \text{image}\, e_{B \times \mathbb{N}} \subset \mathbb{N})\ (T(n) \in \text{image}\, e_{B \times \{\pm\}} \text{ or } T(n) \text{ is undefined.})$$

As $e_{\{\pm\}}$ is injective, T' above is uniquely determined *if* it exists. 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 one construction of elements of $\mathcal{D}_B$, from Turing machines of the following form.

Definition 3.6. *Let $B \in \mathcal{S}$. Define $\mathcal{T}_B$ to be the set of $T \in \mathcal{T}$ s.t. exists $T' : \mathbb{N} \to B \times \{\pm\}$, and a commutative diagram:*

$$\begin{tikzcd} \mathbb{N} \arrow[r, "T'"] \arrow[d, "id"] & B \times \{\pm\} \arrow[d, "e_{B\times\{\pm\}}"] \\ \mathbb{N} \arrow[r, "T"] & \mathbb{N}. \end{tikzcd}$$

From now on, given $T \in \mathcal{T}_B$, if we write T' then it is will be assumed to be of the form above. As before, it is uniquely determined when exists.

Lemma 3.7. *For $(A, e_A) \in \mathcal{S}$ define the map:*

$$e_A^{-1} : \mathbb{N} \to A, \tag{3.8}$$

by

$$e_A^{-1}(n) = \begin{cases} e_A^{-1}(n), & \text{if } n \in \operatorname{image} e_A \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

Then e_A^{-1} is computable.

Proof. Let $T \in \mathcal{T}$ compute the map $\mathbb{N} \to \mathbb{N}$ defined by

$$n \mapsto \begin{cases} n, & \text{if } n \in \operatorname{image} e_A \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

As $\operatorname{image} e_A$ is decidable T does exist.

Then clearly we have a commutative diagram:

$$\begin{tikzcd} \mathbb{N} \arrow[r, "e_A^{-1}"] \arrow[d, "id"] & A \arrow[d, "e_A"] \\ \mathbb{N} \arrow[r, "T"] & \mathbb{N}, \end{tikzcd}$$

so that e_A^{-1} is computable. □

Lemma 3.9. *Let $(B, e_B) \in \mathcal{S}$. There is a computable total map*

$$K = K_{(B,e_B)} : \mathcal{T} \to \mathcal{T},$$

with the properties:

(1) *For each T, $K(T) \in \mathcal{T}_B$.*

(2) *If $T \in \mathcal{T}_B$ then $K(T)$ and T compute the same maps $\mathbb{N} \to B \times \{\pm\}$.*

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

$$\mathcal{T} \times \mathbb{N} \xrightarrow{U} \mathbb{N} \xrightarrow{e_{B\times\{\pm\}}^{-1}} B \times \{\pm\}.$$

Hence, G is a composition of computable maps and so is computable. By Part 3 of the Proposition 2.4, there is an induced computable map $K : \mathcal{T} \to \mathcal{T}$ so that for each T, $K(T)$ computes $G^T : \mathbb{N} \to B \times \{\pm\}$, $G^T(n) = G(T, n)$. By construction, if $T \in \mathcal{T}_B$ then $T' = (K(T))'$. So that we are done. □

3.1.1. *Constructing decision Turing machines.* We will need a few preliminaries as we need to deal with the following issue. For a given computable $\mathbb{N} \to B \times \{\pm\}$, we may construct a total computable map with the same image, but possibly losing the stability conditions. So we must adjust the totalization procedure to keep track of the stability conditions.

The following is well known:

Lemma 3.10. *There is a computable map $Tot : \mathcal{T} \to \mathcal{T}$ s.t. $\forall T \in \mathcal{T}$:*

(1) *$Tot(T)$ is total for each T.*

(2) $\operatorname{image}(Tot(T)) = \operatorname{image} T$.

Let

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

be the natural projections. We say that a subset $S \subset \mathbb{N} \times B \times \{\pm\}$ is *graphical*, if $\pi_{\mathbb{N}}^{-1}(n)$ is at most one element for each $n \in \mathbb{N}$.

For a graphical $S \subset \mathbb{N} \times B \times \{\pm\}$ there is a natural total order, defined as follows. For $s, s' \in S$, $s \leq s'$ if $\pi_{\mathbb{N}}(s) \leq \pi_{\mathbb{N}}(s')$. So a finite graphical $S \subset \mathbb{N} \times B \times \{\pm\}$ determines a list $l_S \in L(B \times \{\pm\})$ by

$$l_S(i) = \pi_{B\times\{\pm\}}(s_i), \tag{3.11}$$

where s_i is the $i-1$ element of S with respect to the total order.

Definition 3.12. *Let $l \in L(B \times \{\pm\})$. Define $b \in B$ to be* **l-stable** *if there is an $m \leq \operatorname{length}(l)$ s.t. $l(m) = (b, +)$ and there is no $m < k \leq \operatorname{length}(l)$ s.t. $l(k) = (b, -)$.*

Given a map $f : \mathbb{N} \to B \times \{\pm\}$ let $\operatorname{gr} f : \mathbb{N} \to \mathbb{N} \times B \times \{\pm\}$ be the map $\operatorname{gr} f(n) = (n, f(n))$. Expanding on Lemma 3.9, there is a total computable map:

$$Gr_B : \mathcal{T} \to \mathcal{T}, \tag{3.13}$$

with the property: $Gr_B(T)$ computes the map $\operatorname{gr}(K_B(T)')$.

Set

$$f^T := e^{-1}_{\mathbb{N}\times B\times\{\pm\}} \circ Tot \circ Gr_B(T) \circ e_{\mathbb{N}}. \tag{3.14}$$

So that $f^T : \mathbb{N} \to \mathbb{N} \times B \times \{\pm\}$ and is total.

Finally, define

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

to be the map:

$$G(b, T, n) = \begin{cases} +, & b \text{ is } l_S\text{-stable, for } S = \operatorname{image}(f^T|_{\{0,\dots n\}}). \\ -, & \text{otherwise.} \end{cases}$$

Lemma 3.15. *G is computable.*

Proof. This is clear, but we make this explicit. Let

$$g : \mathbb{N} \to L(\mathbb{N}) \tag{3.16}$$

be the map $g(n) = \{0, \dots, n\}$, it is clearly computable directly by part 4 of the Proposition 2.4. Let

$$L^{graph}(\mathbb{N} \times B \times \{\pm\}) \subset L(\mathbb{N} \times B \times \{\pm\})$$

consist of lists l s.t. $S = \text{image}\, l$ is graphical. The latter subset is given the induced encoding by restricting $e_{L(\mathbb{N}\times B\times\{\pm\})}$.

Then we can express G as the composition of the sequence of maps:

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

where

(1) $ord : L^{graph}(\mathbb{N} \times B \times \{\pm\}) \to L(B \times \{\pm\})$ is the map $ord(l) = l_S$, for $S = \text{image}\, l$.

(2) The last map is:

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

The latter two maps are computable by explicit verification. In particular all the maps in the composition are computable and so G is computable.

□

Let

$$Dec_B : \mathcal{T} \to \mathcal{T}, \tag{3.17}$$

be the computable total map corresponding G via Axiom 3, so that $Dec_B(T)$ is the Turing machine computing

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

The following is immediate from the construction:

Lemma 3.18. *$Dec_B(T)$ has the properties:*

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

(2) *$Dec_B(T)$ is total.*

Definition 3.19. *For 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 $T \in \mathcal{T}_B$ we say that b is T-**stable** if it is T'-stable in the sense of Definition 3.1.*

Lemma 3.20. *Suppose that $T \in \mathcal{T}_B$ 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 $T(m) = (b, +)$ and there is no $n > m$ so that $T(m) = (b, -)$. By construction, there exists an m' s.t. $f^T(m') = (m, b, +)$. It is then immediate that b is l_{S_n}-stable, for any $S_n = \text{image}(f^T|_{\{0,\ldots,n\}})$ s.t. $n \geq m'$.

Then by construction of G, $G(b, T, m') = +$ and there is no $n > m'$ s.t. $G(b, T, m') = -$. Thus b is G^T-decided and so $Dec_B(T)$-decided.

The converse is also clear. Suppose b is G^T decided. Let N be s.t.

(1) $G(b, T, N) = +$.

(2) There is no $n > N$ s.t. $G(b, T, n) = -$.

By property 1 there is an $n_0 \in \{0, \dots, N\}$ s.t. $f^T(n_0) = (n_b, b, +)$, and there is no $n \in \{0, \dots, N\}$, with $pr_{\mathbb{N}} \circ f^T(n) > n_b$, s.t. $pr_{B \times \{\pm\}} \circ f^T(n) = (b, -)$.

Now, if b is not T-stable then there is a $n' > n_b$ s.t. $T(n') = (b, -)$. By the previous paragraph,

$$n' \notin \operatorname{image} pr_{\mathbb{N}} \circ f^T|_{\{0,\dots,N\}}.$$

So there is an $n > N$ so that $f^T(n) = (n', b, -)$. But then $G(b, T, n) = -$ which contradicts property 2. □

Example 3.21. By the Example 3.3 above there is a computable map

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

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

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

Definition 3.22. *Given a pair of maps*

$$M_0 : B \times \mathbb{N} \to \{\pm\}$$
$$M_1 : B \times \mathbb{N} \to \{\pm\},$$

we say that they are **stably equivalent** *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' is stably equivalent to M.*

3.2. **Decision maps in first order theories.** Let $\mathcal{L}$ be as in the Introduction. Let $\mathcal{T}_{\mathcal{L}}$ be as in Definition 3.6 with respect to $B = \mathcal{L}$. The following is a version for stably c.e. theories of the classical fact, going back to 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 computably constructive.

Notation 1. Note that each $T \in \mathcal{T}_{\mathcal{L}}$, determines the set

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

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

Lemma 3.23. *Let $\mathcal{L}$ be given. There is a computable total map:*

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

so that $\forall T \in \mathcal{T} : C(T) \in \mathcal{T}_{\mathcal{L}}$. If in addition $T \in \mathcal{T}_{\mathcal{L}}$ then $(C(T))^s$ is the deductive closure of T^s.

Proof. Let $L(\mathcal{L})$ be the list construction on $\mathcal{L}$ as previously. Let $\Phi : L(\mathcal{L}) \times \mathbb{N} \to \mathcal{L}$ be as in the Axiom 5.

Using the map *ord* from Lemma 3.15, define a map

$$\zeta : L(\mathcal{L}) \times L^{graph}(\mathbb{N} \times \mathcal{L} \times \{\pm\}) \to \{\pm\}$$

by

$$\zeta(l, l') = \begin{cases} +, & \text{if for each } 0 \le i \le \text{length}(l),\ l(i) \text{ is } ord(l')\text{-stable.} \\ -, & \text{otherwise.} \end{cases}$$

Utilizing Proposition 2.4 we readily see that ζ is computable.

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

$$\mathcal{T} \times L(\mathbb{N}) \xrightarrow{Gr_{\mathcal{L}} \times id} \mathcal{T} \times L(\mathbb{N}) \xrightarrow{LU} L(\mathbb{N}) \xrightarrow{L(e^{-1}_{\mathbb{N}\times\mathcal{L}\times\{\pm\}})} L^{graph}(\mathbb{N} \times \mathcal{L} \times \{\pm\}).$$

All the maps in the composition are computable directly by the Proposition 2.4 and Lemma 3.7 and so H 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

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

Let

$$pr_{\mathcal{L}} : \mathbb{N} \times \mathcal{L} \times \{\pm\} \to \mathcal{L}$$

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

$$U^T_{n+1} := U^T_n \cup \bigcup_{l \in L_{n+1}(\mathbb{N})} \bigcup_{0 \leq m \leq n+1} (\Phi(L_{pr_{\mathcal{L}}} \circ H(T,l), m), \zeta(L_{pr_{\mathcal{L}}} \circ H(T,l), H(T, \{0, \ldots, n+1\})).$$

As in Example 3.3 we define

$$U^T : \mathbb{N} \to \mathcal{L} \times \{\pm\}, \quad U^T(n) := U^T_{n+1}(n).$$

And this induces a total map

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

$U(T,n) := U^T(n)$. U is computable by explicit verification, utilizing Proposition 2.4, and the recursive program for $\{U^T_n\}$. Hence, by part 3 of Proposition 2.4, there is an induced by U computable total map:

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

s.t. for each $T \in \mathcal{T}$, $C(T)$ computes U^T.

By construction, this has the needed properties, and we are done. □

Notation 2. Given $M : \mathbb{N} \to \mathcal{L} \times \{\pm\}$, and some translation $i : \mathcal{A} \to \mathcal{L}$, we use from now the shorthand: $M \vdash^i \alpha$, for $M^s \vdash^i \alpha$, whenever $\alpha \in \mathcal{A}$, and whenever this may cause no confusion.

Let $\mathcal{F}_0$, as in the introduction, denote the set of formulas ϕ of arithmetic with one free variable so that for any term n, $\phi(n)$ is Σ^0_0 and in particular is Q-decidable.

Definition 3.24. *Let $\mathcal{L}$ be a first order language, with some translation $i : \mathcal{A} \to \mathcal{L}$. We say that $M : \mathbb{N} \to \mathcal{L} \times \{\pm\}$ is* **speculative** *(with respect to i) if the following holds. Let $\phi \in \mathcal{F}_0$, and set*

$$\alpha_\phi = \forall m \ \phi(m), \tag{3.25}$$

then

$$(\forall m \ Q \vdash \phi(m)) \implies M \vdash^i \alpha_\phi.$$

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 3.26. *Let $\mathcal{L}$ be given and $i : \mathcal{A} \to \mathcal{L}$ be some fixed 2-translation. Then there is a computable total map $Spec_i : \mathcal{T} \to \mathcal{T}$, with the following properties:*

(1) image $Spec_i \subset \mathcal{T}_{\mathcal{L}}$.

(2) *Suppose that $T \in \mathcal{T}_{\mathcal{L}}$ and $T^s \vdash^i Q$, set $T_{spec} = Spec_i(T)$ then T'_{spec} is speculative, moreover if T' is total then so is T'_{spec}.*

(3) *Using Notation 1, if $T \in \mathcal{T}_{\mathcal{L}}$ then $T^s_{spec} \supset T^s$*

(4) *If $T \in \mathcal{T}_{\mathcal{L}}$ and T^s is 1-consistent with respect to i then T^s_{spec} is consistent.*

Proof. $\mathcal{F}_0$ is 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. Let $G \subset \mathcal{F}_0$ be the subset of formulas ϕ s.t. $\forall n \; Q \vdash \phi(n^\circ)$, where n° denotes the corresponding numeral.

We then need:

Lemma 3.27. *There is a total computable map $J : \mathbb{N} \to \mathcal{F}_0 \times \{\pm\}$ with the property:*

$$J^s = G.$$

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

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

In what follows each J_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 $\{J_n\}_{n \in \mathbb{N}}$ recursively by $J_0 := \emptyset$,

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

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

We set $J(n) := J_{n+1}(n)$. This is a total map

$$J : \mathbb{N} \to \mathcal{F}_0 \times \{\pm\},$$

having the property $F(\mathbb{N}) = \cup_n \text{image}(J_n)$. J is computable by explicit verification, using Proposition 2.4. □

Returning to the proof of the theorem. Let $K = K_{\mathcal{L}} : \mathcal{T} \to \mathcal{T}$ be as in Lemma 3.9. For $\phi \in \mathcal{F}_0$ let α_ϕ be as in (3.25). Define: $H : \mathcal{T} \times \mathbb{N} \to \mathcal{L} \times \{\pm\}$ by

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

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 by Proposition 2.4. (Factor H as a composition of computable maps as previously.)

Let $Spec_i : \mathcal{T} \to \mathcal{T}$ be the computable map corresponding to H via Axiom 3. In particular, for each $T \in \mathcal{T}$, $Spec_i(T)$ computes the map

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

which by construction is speculative. Now, $Spec_i(T)$ satisfies the Properties 1, 2, 3 immediately by construction.

It only remains to check Property 4. In what follows, let $T \in \mathcal{T}_{\mathcal{L}}$, and set $F_{spec} = (T'_{spec})^s$, and $F = (T')^s$, which is 1-consistent by assumption.

Lemma 3.28. *F_{spec} consistent unless for some $\phi \in G$*

$$F \vdash^i \neg \forall m \; \phi(m).$$

Proof. Suppose that F_{spec} is inconsistent so that:

$$F \cup \{i(\alpha_{\phi_1}), \ldots, i(\alpha_{\phi_n})\} \vdash \alpha \wedge \neg\alpha$$

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

$$F \vdash^i \neg(\alpha_{\phi_1} \wedge \ldots \wedge \alpha_{\phi_n}), \text{ as } i \text{ preserves logical operators.}$$

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 F_{spec} inconsistent, then by the lemma above for some $\phi \in G$:

$$F \vdash^i \exists m\ \neg\phi(m).$$

By assumption F is 1-consistent with respect to i and so:

$$\exists m\ F \nvdash^i \phi(m).$$

But ϕ is in G, and $F \vdash^i Q$ (part of our 1-consistency assumption) so that $\forall m\ F \vdash^i \phi(m)$ and so

$$\exists m\ F \vdash \neg\phi(m) \wedge \phi(m).$$

So F is inconsistent, a contradiction, so F_{spec} is consistent. □

4. The stable halting problem

We now discuss a version of the halting problem in the context of stable computability. This will illustrate the main ideas that will be used in the proof of the incompleteness theorems, and will allow us to setup notation for later use.

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

Lemma 4.1. *There is a computable total map*

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

with the properties:

(1) *For each T, $\Omega(T) \in \mathcal{D}_{\mathcal{T}}$.*

(2) *If $T \in \mathcal{T}_{\mathcal{D}}$ then $\Omega(T)$ and T compute the same maps $\mathcal{T} \times \mathbb{N} \to \{\pm\}$, that is $T' = (\Omega(T))'$.*

Proof. Analogous to the proof of Lemma 3.9. □

Definition 4.2. *For $T \in \mathcal{D}_{\mathcal{T}}$, T* **is T-decided**, *is a special case of Definition 3.19. 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.*

In what follows, denote by $s(T)$ the sentence:

$$T \text{ is not } \Omega(T)\text{-decided.}$$

which can be naturally interpreted as a sentence of arithmetic, but we leave this implicit for the moment.

Definition 4.3. *We say that* $D : \mathcal{T} \times \mathbb{N} \to \{\pm\}$ *is* **stably sound** *if*

$$\forall T \in \mathcal{T}\ (T \text{ is } D\text{-decided}) \implies s(T).$$

We say that D **stably decides** T *if:*

$$s(T) \implies T \text{ is } D\text{-decided}.$$

We say that D *is* **stably sound and complete** *if* D *is stably sound and* D *stably 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" the sentence $s(T)$ is true, at the moment n.
- For each T, n $D(T, n) = -$ if and only if D cannot "decide" the sentence $s(T)$ at the moment n, or D "decides" that $s(T)$ is false.

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

$$T \text{ stably computes } D,$$

where this is as in Definition 3.22.

Lemma 4.4. *If* D *is stably sound then*

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

Proof. If T is D-decided then since D is stably sound, T is not $\Omega(T)$-decided. So if in addition $\Theta_{D,T}$ then by property 2 of Ω, $(\Omega(T))' = D$. And so T is not D-decided a contradiction. □

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

Theorem 4.5. *There is no (stably) computable* $D : \mathcal{T} \times \mathbb{N} \to \{\pm\}$ *that is stably sound and complete.*

Proof. Let D be stably sound and complete. Then by the above lemma we obtain:

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

Again as a consequence of property 2 of Ω, it is immediate:

$$(\forall T \in \mathcal{T})\ (\Theta_{D,T} \implies (\neg(T \text{ is } D\text{-decided})) \implies \neg(T \text{ is } \Omega(T)\text{-decided}))\,. \tag{4.7}$$

So combining (4.6), (4.7) above we obtain

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

But D is complete so

$$(T \text{ is } \Omega(T)\text{-decided}) \implies T \text{ is } D\text{-decided}$$

and so:

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

Combining with (4.6) we get

$$(\forall T \in \mathcal{T})\ \neg\Theta_{D,T},$$

which is what we wanted to prove. □

5. Incompleteness theorems

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

$$H(F, T, n) := (Dec_{\mathcal{L}}(C(Spec_i(F))))'(i \circ s(T), n).$$

We can express H as the composition of the sequence of maps:

$$\mathcal{T} \times \mathcal{T} \times \mathbb{N} \xrightarrow{(Dec_{\mathcal{L}} \circ C \circ Spec_i) \times (i \circ s) \times id} \mathcal{T} \times \mathcal{L} \times \mathbb{N} \xrightarrow{id \times e_{\mathcal{L} \times \mathbb{N}}} \mathcal{T} \times \mathbb{N} \xrightarrow{U} \mathbb{N} \xrightarrow{e^{-1}_{\{\pm\}}} \{\pm\}. \tag{5.1}$$

Thus, H is a composition of maps that are computable by Proposition 2.4 and so H is computable. Hence, by part 3 of Proposition 2.4, there is an associated total computable map:

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

s.t. for each $F \in \mathcal{T}$, $Tur(F)$ computes the map

$$D^F : \mathcal{T} \times \mathbb{N} \to \{\pm\},\ D^F(T, n) = H(F, T, n), \tag{5.3}$$

In what follows, $T \in \mathcal{T}_{\mathcal{L}}$, and we rename T' as M so $M : \mathbb{N} \to \mathcal{L} \times \{\pm\}$.

Notation 3. As usual, for a map $J : \mathbb{N} \to \mathcal{L} \times \{\pm\}$, notation of the form $J \vdash \alpha$ means $J^s \vdash \alpha$. If $i : \mathcal{A} \to \mathcal{L}$ is a translation, then $J \vdash^i \alpha$ will denote $J^s \vdash^i \alpha$. J is (n)-consistent stands for J^s is (n)-consistent.

Proposition 5.4. *For (M, T) as above and given a 2-translation $i : \mathcal{A} \to \mathcal{L}$ we have:*

1. *M is 1-consistent relative to the translation $i \implies M \nvdash i \circ s(Tur(T))$.*
2. *M is 2-consistent relative to the translation $i \implies M \nvdash \neg i \circ s(Tur(T))$.*
3. *M is 1-consistent relative to the translation $i \implies s(Tur(T))$.*

Proof. Since T will be fixed, let us abbreviate $s(Tur(T))$ by $\mathfrak{s}$. Set $N := (C \circ Spec_i(T))'$, in particular this is a speculative (with respect to i) map $\mathbb{N} \to \mathcal{L} \times \{\pm\}$. Suppose that $M \vdash i(\mathfrak{s})$. In particular, $i(\mathfrak{s})$ is N-stable, and so by Lemma 3.20 $i(\mathfrak{s})$ is $Dec_{\mathcal{L}}(C(Spec_i(T)))$-decided. And so $Tur(T)$ is D^T-decided by definitions, where D^T is as in (5.3).

Now, since $Tur(T)$ computes D^T by construction, we have $(\Omega(Tur(T)))' = D^T$ and so:

$$\begin{aligned} Tur(T) \text{ is } D^T\text{-decided} &\iff Tur(T) \text{ is } \Omega(Tur(T))\text{-decided} \\ &\iff \neg\mathfrak{s}. \end{aligned} \tag{5.5}$$

That it to say:

$$(M \vdash i(\mathfrak{s})) \implies \neg\mathfrak{s}. \tag{5.6}$$

Now, $\neg\mathfrak{s}$ is an arithmetic sentence of the form:

$$\exists m \forall n\ \gamma(m, n), \tag{5.7}$$

where $\gamma \in \Sigma^0_0$.

And we have:

$$\neg\mathfrak{s} \implies (\exists m \forall n)\ Q \vdash \gamma(m, n), \quad \text{since } \gamma \text{ is } \Sigma^0_0 \tag{5.8}$$

$$\implies \exists m\ N \vdash^i \forall n\ \gamma(m, n), \quad \text{since } N \text{ is speculative} \tag{5.9}$$

$$\implies N \vdash^i \neg\mathfrak{s}, \text{ (by existential introduction).} \tag{5.10}$$

And so combining with (5.6), we get:

$$(M \vdash i(\mathfrak{s})) \implies (N \vdash i(\mathfrak{s})) \wedge (N \vdash \neg i(\mathfrak{s})).$$

Since by Theorem 3.26

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

it follows:

$$M \text{ is 1-consistent} \implies M \nvdash^i \mathfrak{s} \tag{5.11}$$

$$\implies (Tur(T) \text{ is not } \Omega(Tur(T))\text{-decided}), \text{ by (5.5)} \tag{5.12}$$

$$\implies \mathfrak{s}. \tag{5.13}$$

This proves the first and the third part of the proposition.

Now suppose

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

Set

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

where $\gamma(m,n)$ is as in (5.7). Now,

$$\begin{aligned} M \vdash^i \neg\mathfrak{s} &\iff M \vdash^i \exists m\ \phi(m) \\ &\implies \exists m\ M \nvdash^i \neg\phi(m), \quad \text{by 2-consistency} \\ &\implies \exists m \forall n\ Q \vdash \gamma(m,n), \quad \text{as } M \vdash^i Q \text{ and } \gamma(m,n) \text{ is } Q\text{-decidable.} \end{aligned}$$

And so,

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

Now,

$$\neg\mathfrak{s} \implies N \vdash \mathfrak{s},$$

by definitions and (5.5). So:

$$\begin{aligned} (M \text{ is 2-consistent}) \wedge (M \vdash^i \neg\mathfrak{s}) &\implies N \vdash \mathfrak{s} \\ &\implies N \text{ is inconsistent} \\ &\implies M \text{ is not 1-consistent, by Theorem 3.26} \\ &\implies M \text{ is not 2-consistent.} \end{aligned}$$

So we get a contradiction, and so:

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

This finishes the proof of the proposition. □

Proof of Theorem 1.6. The computable map $\mathcal{G}$ is defined to be $T \mapsto s(Tur(T))$. Then the theorem follows immediately by the proposition above. □

Proof of Corollary 1.8. Suppose that

$$(F \vdash^i F \text{ is 1-consistent}) \wedge (F \vdash^i T \text{ stably enumerates } F) \wedge (F \text{ is 1-consistent})$$

then since $F \vdash^i ZFC$,

$$(F \vdash s(Tur(T))) \wedge F \text{ is 1-consistent}$$

by part three of Proposition 5.4. But this contradicts part one of Proposition 5.4. □

Proof of Theorem 1.4. Let M be a standard model of $F_{i,\mathcal{Z}}$ as in the definition of strong consistency. In particular:

$$(\forall \alpha \in F_{i,\mathcal{Z}})\ M \models \alpha. \tag{5.14}$$

Suppose that $F \vdash^i F$ is 1-consistent, then $M \models (F \vdash^i F$ is 1-consistent), using that M is standard. Suppose also:

$$F \vdash^i (\exists T \in \mathcal{T})\ T \text{ stably computes } F,$$

then by (5.14)

$$M \models (\exists T \in \mathcal{T})\ T \text{ stably computes } F.$$

And so we obtain:

$$M \models (F \text{ is 1-consistent}) \wedge ((\exists T \in \mathcal{T})\ T \text{ stably computes } F) \wedge (F \vdash^i F \text{ is 1-consistent}),$$

that is

$$M \models (\exists T \in \mathcal{T}_{\mathcal{L}})\ ((T')^s \text{ is 1-consistent}) \wedge (T' \vdash^i (T')^s \text{ is 1-consistent})). \tag{5.15}$$

Rephrasing Corollary 1.8 we get:

$$ZFC \vdash (\forall T \in \mathcal{T}_{\mathcal{L}})\ ((T')^s \text{ is 1-consistent} \implies T' \nvdash^i (T')^s \text{ is 1-consistent}).$$

And $F \vdash ZFC$ and so again by (5.14) we get:

$$M \models \neg((\exists T \in \mathcal{T}_{\mathcal{L}})\ ((T')^s \text{ is 1-consistent} \wedge T' \vdash^i (T')^s \text{ is 1-consistent})),$$

but this contradicts (5.15). □

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

We now give some partly physical motivation for the theory above, in particular explaining why stable computability and abstract languages were important for us. We aim to be very brief, as this is an excursion. But developing this appendix would be very interesting in an appropriate venue.

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, it is well known (see for instance [3]) that solutions of fluid flow and N-body problems are generally non Turing computable (over $\mathbb{Z}$, and probably over $\mathbb{R}$ cf. [2]) as modeled in mathematics of classical mechanics. But in a more physically accurate and fundamental model both of the processes above may become computable.

The question posed by Turing [1], 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, pg. 310], relating the question to existence of absolutely unsolvable Diophantine problems, see also Feferman [5], and Koellner [8], [9] for a discussion.

We now discuss the question from the perspective of our main results. First by an idealized mathematician, we mean a mapping $H : \mathbb{N} \to \mathcal{L} \times \{\pm\}$, with $\mathcal{L}$ a first order language. The language $\mathcal{L}$ is meant to be chosen so that it is sufficient to formalize physical laws (we don't need to formalize everything, the context will be self apparent). This might be the language of set theory, but perhaps one needs more. [2] The mapping H is meant to be the actual time stamped output of a mathematician, idealized so that their brain does not deteriorate in time, see also Remark 1.2.

[2] It is of course possible that no such language exists, but this is not our expectation: mathematics is unreasonably effective at formalizing the universe.

We set $\mathcal{H} = H^s$. Now, soundness and in particular strong consistency (Definition 2.15) of the stabilization $\mathcal{H}$ is not an unreasonable hypothesis for our "idealized" mathematician, as mathematical knowledge does appear to stabilize on truth. Here the word 'stabilize' is used in the standard English language sense, but in this setting this is equivalent to the soundness of the mathematical stabilization H^s.

Without delving deeply into interpretations, we suppose the following axioms for $\mathcal{H}$.

(1) $\mathcal{H}$ is $\mathcal{L}$-definable. Meaning, that it is definable with respect to some $\mathcal{L}$-structure (informally, the latter is obtained from some mathematical (sub)model of the physical universe). [3]

(2) $\mathcal{H} \vdash^i ZFC$, where $i : \mathcal{Z} \to \mathcal{L}$ is a 2-translation, recall the Definition 2.12.

(3) The "Penrose property" holds:

$$\mathcal{H} \vdash^i (\mathcal{H} \text{ is 1-consistent.})$$

The "Penrose property" is motivated by ideas of Roger Penrose as appearing in references above. This property makes sense if our idealized mathematician knows the definition of $\mathcal{H}$, and asserts their soundness, and hence 1-consistency of $\mathcal{H}$. That the definition of $\mathcal{H}$ is known, is not unlikely. The idea for this is to map the brain (synapses, and other relevant fine structure); then assuming one knows the working of all underlying physical processes, use this to reconstruct the $\mathcal{L}$-theoretic definition of $\mathcal{H}$. This would be a fantastically difficult thing to do, but theoretically possible. Given this, there is no obvious reason to reject the above axioms.

The following just paraphrases Theorem 1.4.

Theorem A.1. *For any $\mathcal{H}, \mathcal{L}$ as above, one of the following holds:*

(1) *$\mathcal{H}$ is not strongly consistent.*

(2) *$\mathcal{H} \nvdash^i \mathcal{H}$ is stably c.e.. (In particular, given our interpretation of $\mathcal{H}$ as representing an idealized mathematician, it is unable to disprove existence of absolutely non Turing computable physical processes.)*

Acknowledgements. Peter Koellner for helpful discussions on related topics.

References

[1] A.M. Turing, *Computing machines and intelligence*, Mind, 49 (1950), pp. 433–460.
[2] 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.
[3] E. P.-S. D. P. F. Cardona, Robert; Miranda, *Constructing turing complete euler flows in dimension 3*, PNAS, 118 (2021).
[4] S. Feferman, *1-consistency and faithful interpretations*, 6 (1962).
[5] ———, *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] R. I. Soare, *Turing computability. Theory and applications*, Berlin: Springer, 2016.

[3] This is natural: as H is determined by some physical processes, it is $\mathcal{L}$-definable, since by assumption the physical laws are stated in the language $\mathcal{L}$.

University of Colima, Department of Sciences, CUICBAS

Email address: yasha.savelyev@gmail.com