Very Weak, Essentially Undecidabile Set Theories* **

Domenico Cantone[1][0000−0002−1306−1166],
Eugenio G. Omodeo[2][0000−0003−3917−1942], and
Mattia Panettiere[1][0000−0002−9218−5449]

[1] Dept. of Mathematics and Computer Science, University of Catania, Italy
domenico.cantone@unict.it, mattia.panettiere@gmail.com
[2] Dept. of Mathematics and Earth Sciences, University of Trieste, Italy
eomodeo@units.it

Abstract. In a first-order theory Θ, the decision problem for a class of formulae Φ is solvable if there is an algorithmic procedure that can assess whether or not the existential closure $\varphi^{\exists}$ of φ belongs to Θ, for any $\varphi \in \Phi$. In 1988, Parlamento and Policriti already showed how to apply Gödel-like arguments to a very weak axiomatic set theory, with respect to the class of Σ_1-formulae with $(\forall\exists\forall)_0$-matrix, i.e., existential closures of formulae that contain just restricted quantifiers of the kind $(\forall x \in y)$ and $(\exists x \in y)$ and are writeable in prenex form with at most two alternations of restricted quantifiers (the outermost quantifier being a '$\forall$'). While revisiting their work, we show slightly stronger theories under which incompleteness for recursively axiomatizable extensions holds with respect to existential closures of $(\forall\exists)_0$-matrices, namely formulae with at most one alternation of restricted quantifiers.

Keywords: Decidability · Set Theory · Gödel Incompleteness.

Introduction

One often resorts to meta-level reasoning within a formal system, in order to support meta-mathematical investigations (e.g., concerning syntactic boundaries beyond which the decision problem for an axiomatic theory becomes algorithmically unsolvable), meta-programming in declarative languages [4], or agent-based explainable AI applications (if the agents are to exhibit self-awareness of any form [2]).

The resources that a first-order theory must provide to make meta-level reasoning doable at all are surprisingly simple. In the realm of number theory, a

** We gratefully acknowledge partial support from project "STORAGE—Università degli Studi di Catania, Piano della Ricerca 2020/2022, Linea di intervento 2", and from INdAM-GNCS 2019 and 2020 research funds.

minimal arithmetic (extremely weak relative to Peano's arithmetic) was proposed by Raphael M. Robinson in [11]; in the realm of set theory, an even simpler axiomatic endowment (only consisting of the null-set axiom along with an axiom enabling the adjunction of a single element to a set) was proposed by Robert L. Vaught in [13]. In either case, the proposed axiom system constitutes an *essentially undecidable theory*, i.e., a theory none of whose consistent axiomatic extensions has an algorithmically solvable derivability problem—as can be proved *à la* Gödel.

Vaught's result can be improved in at least two ways: (1) by making all steps needed for the so-called 'arithmetization of syntax' task more transparent; (2) by basing such a task on formulae of an extremely low syntactic structure. Franco Parlamento and Alberto Policriti contributed to these two amelioration goals [10,9], *referring as a yardstick for measuring syntactical complexity to the number of quantifier alternations in the lowest level of Azriel Lévy's hierarchy of set-theoretic formulae* [6]. The axiom system that they exploited was still finite but slightly broader than the one by Vaught.

This paper further develops the techniques by Parlamento and Policriti, by broadening the axiomatic system even further in order to achieve greater transparency and to reduce by one the number of quantifier alternations.

The decision problem for a class of formulae Φ of the language of a given theory Θ—a consistent axiomatic *set theory* in the ongoing—is said to be *solvable* when there is an algorithmic procedure that, taken in input any $\varphi \in \Phi$ with n free variables $x_1, x_2, \ldots, x_n$, establishes whether or not $\Theta \vdash \exists x_1 \exists x_2 \cdots \exists x_n \, \varphi$ holds, and outputs: true if things are so, false if $\Theta \nvdash \exists x_1 \exists x_2 \cdots \exists x_n \, \varphi$ (in particular if $\Theta \vdash \neg \exists x_1 \exists x_2 \cdots \exists x_n \, \varphi$). While studying (un)decidability in fragments of set theory, it is worth considering *restricted quantifiers*, i.e., quantifiers of the form:

$$\begin{array}{ll} (\forall x \in y)\varphi & \overset{\text{Def}}{\longleftrightarrow} \quad \forall x(x \in y \ \rightarrow \ \varphi)\,, \\ (\exists x \in y)\varphi & \overset{\text{Def}}{\longleftrightarrow} \quad \exists x(x \in y \ \wedge \ \varphi)\,. \end{array} \tag{1}$$

When a formula contains only restricted quantifiers, it is called a Δ_0-formula (see [6]). Furthermore, if it is logically equivalent to some prenex formula with n alternating sequences of unbounded quantifiers in the prefix starting with universal quantifiers, then it is called $(\forall\exists\forall \cdots Q_n)_0$ (where Q_i is $\forall$ when i is odd, and $\exists$ otherwise). Clearly, a complete theory is decidable; but the converse also holds when we restrict ourselves to consistent classes of existential closures of Δ_0-formulae. In [10], it has been investigated how, taking the very weak set theory T_0 comprising extensionality, null-set axiom, single-element addition and removal axioms, an argument akin to the ones leading to Gödel incompleteness theorems can be applied to the class of $(\forall\exists\forall)_0$-formulae. Thanks to those arguments, it is possible to show that every recursively axiomatizable extension Θ of T_0 is incomplete with respect to the class of $(\forall\exists\forall)_0$-formulae, and therefore the decision problem for that class is undecidable in every extension Θ of T_0. Since the base theory T_0 is extremely elementary, the argument applies to every reasonable set theory. The limit found in [10] on the Δ_0-complexity of the class

of formulae seems to be rather tight, with no room for improvement in that direction. Nevertheless, by slightly expanding the axiomatic core, we have found a way to lower that limit under suitable conditions. Indeed, we consider extensions of T_0 that include the axiom of foundation and prove that if the concept of "being a natural number" is expressible by a $(\forall\exists)_0$-formula, then the incompleteness arguments can be generalized with respect to the whole class of $(\forall\exists)_0$-formulae. At the end we will cite two examples that allow for such arguments, expanding the core theory with either the separation axiom schema or an axiom stating that every set is (hereditarily) finite.

1 A succinct axiomatic endowment for Set Theory

We will consider the first-order language $\mathcal{L}_\in$ endowed with:

- an infinite supply $\nu_0, \nu_1, \nu_2, \ldots$ of set variables;
- two dyadic relators $\in, =$, designating membership and equality, respectively, as the only predicate symbols of the language;
- the propositional connectives $\neg$ (monadic) and $\to$ (dyadic), designating respectively: negation and material implication;
- the existential quantifier $\exists\,\nu_i$ and the universal quantifier $\forall\,\nu_i$, associated with each set variable ν_i .

The *combinatorial core*, dubbed T, of the axiomatic set theories we will consider consists of the following five postulates:

Extensionality	**(E)**	$\forall\, x\, \forall\, y\, \exists\, v \left(\left(v \in x \leftrightarrow v \in y \right) \to x = y \right),$
Null set	**(N)**	$\exists\, z\, \forall\, v \left(\neg v \in z \right),$
Adjunction	**(W)**	$\forall\, x\, \forall\, y\, \exists\, w\, \forall\, v \left(v \in w \leftrightarrow \left(v \in x \vee \quad v = y \right) \right),$
Removal	**(L)**	$\forall\, x\, \forall\, y\, \exists\, \ell\, \forall\, v \left(v \in \ell \leftrightarrow \left(v \in x \wedge \neg v = y \right) \right),$
Regularity	**(R)**	$\forall\, x\, \exists\, v\, \forall\, y \left(y \in x \to \left(v \in x \wedge \neg y \in v \right) \right).$

In the light of axiom (**E**), stating that distinct sets cannot have the same elements, axiom (**N**) ensures that exactly one empty set exist. Axiom (**W**) and axiom (**L**) induce the two natural operations $(x, y) \overset{\texttt{with}}{\mapsto} x \cup \{y\}$ and $(x, y) \overset{\texttt{less}}{\mapsto} x \backslash \{y\}$. Axiom (**R**) states that every non-empty set x has an element v that does not intersect x; in synergy with (**N**) and (**W**), it ensures that '$\in$' forms no cycles. On occasions we will consider extending T with further axioms: a theory we will consider is T_f, whose intended universe encompasses *no infinite sets* (cf. [12]); another one is T_s, enhancing T with the *separation* axiom schema (cf. [5]).

2 Ordered pairs, functions, and natural numbers

At the core of our definitional machinery lie the definitions of (un)ordered pair and (un)ordered functions. Indeed, most of the definitions hinge on an extensive use of these set-theoretic structures to lower their Δ_0-complexity. Consider, e.g.,

the variant $\langle x, y\rangle \coloneqq \{x, \{x, y\}\}$ of Kuratowski's classical ordered pair definition, which one can adopt under **(N)**, **(W)**, and **(R)**. This is problematic since, while the extraction of the first component $\pi_1(p)$ of such a pair p is specifiable by means of an $(\exists)_0$-formula, the extraction of the second projection $\pi_2(p)$ calls for a formula with at least one quantifier alternation:

$$y = \pi_2(p) \quad \overset{\text{Def}}{\longleftrightarrow} \quad (\exists x \in p)(\exists q \in p)\big(x \in q \ \wedge\ y \in q \ \wedge\ (\forall z \in q)(z = x \ \vee\ z = y)\big).$$

We rely on a definition that works under **(E)**, **(N)**, **(W)**, and **(L)**—foundation is not required—introduced in [3]. We use the binary operator @ to construct *quasi-pairs*. These will be used in the formation of ordered pairs:

$$\begin{aligned} x@y &\coloneqq \big\{x \texttt{ less } y\,,\ x \texttt{ with } y\big\}\,,\\ \langle x, y\rangle &\coloneqq (x@y)@x\,. \end{aligned}$$

Definition 1. Quasi-pairs *and* ordered pairs *are characterized by the following* $(\forall\exists)_0$*-conjunctions:*

$$\begin{aligned} \mathsf{QPair}(q) \ &\overset{Def}{\longleftrightarrow}\ (\exists\, u \in q)(\exists\, v \in q)\ \neg\, u = v \ \wedge\\ &\quad\ (\forall\, u, v \in q)(\forall\, t \in u)(\forall\, x \in v)\ (x \in u \vee t \in v) \ \wedge\\ &\quad\ (\forall\, u, v \in q)(\forall\, x, z \in v)\ (x \in u \vee z \in u \vee x = z)\,,\\ \mathsf{OPair}(p) \ &\overset{Def}{\longleftrightarrow}\ \mathsf{QPair}(p) \ \wedge\ (\exists\, d \in p)(\exists\, u, v \in d)\neg\, u = v \qquad\qquad \wedge\\ &\quad\ (\forall\, d \in p)(\forall\, u, v \in d)(\forall\, t \in u)(\forall\, y \in v)(y \in u \vee t \in v) \ \wedge\\ &\quad\ (\forall\, d \in p)(\forall\, u, v \in d)(\forall\, y, z \in v)(y \in u \vee z \in u \vee y = z)\,. \end{aligned}$$

We use the first projection extraction on quasi-pairs to obtain the components of ordered pairs, aka 2-tuples, through these $(\exists)_0$-specifications:[3]

$$\begin{aligned} x = \pi^2_1(p) \quad &\overset{\text{Def}}{\longleftrightarrow} \quad (\exists\, u \in p)(\exists\, v \in p)(\ x \in v \wedge \neg\, x \in u\ )\,,\\ y = \pi^2_2(p) \quad &\overset{\text{Def}}{\longleftrightarrow} \quad (\exists\, d \in p)\ y = \pi^2_1(d). \end{aligned}$$

n-tuples and their projections π^n_i $(0 < i \leq n)$, specified as usual in terms of 2-tuples for any n, can thus be captured by $(\forall\exists)_0$- and $(\exists)_0$-formulae, resp.; e.g.:

$$\begin{aligned} \mathsf{Triple}(t) \ &\overset{\text{Def}}{\longleftrightarrow}\ \mathsf{OPair}(t) \ \wedge\ (\forall v_1 \in t)(\forall v_2 \in v_1)(\forall s \in v_2)(s = \pi^2_2(t) \to \mathsf{OPair}(s)),\\ x = \pi^3_1(t) \ &\overset{\text{Def}}{\longleftrightarrow}\ x = \pi^2_1(t),\\ y = \pi^3_2(t) \ &\overset{\text{Def}}{\longleftrightarrow}\ (\exists v_1 \in t)(\exists v_2 \in v_1)(\exists s \in v_2)(s = \pi^2_2(t) \ \wedge\ y = \pi^2_1(s)),\\ z = \pi^3_3(t) \ &\overset{\text{Def}}{\longleftrightarrow}\ (\exists v_1 \in t)(\exists v_2 \in v_1)(\exists s \in v_2)(s = \pi^2_2(t) \ \wedge\ z = \pi^2_2(s)). \end{aligned}$$

Functions. Letting $\in_1 \ \equiv\ \in$, it will be handy to make use of the following recursive definition of $\in_n$, for $n \geq 1$ and for every variable y and formula φ:

$$(\forall x \in_{n+1} y)\varphi \ \overset{\text{Def}}{\longleftrightarrow}\ (\forall z \in y)(\forall x \in_n z)\varphi\,.$$

[3] In specifying projections, it would be pointless to insist that the argument must be an OPair.

We can define functions in the classical way, namely as suitable sets of ordered pairs; their specification is straightforward:

$$\begin{aligned}\mathsf{Fun}(f) \;\overset{\text{Def}}{\longleftrightarrow}\; & (\forall p \in f)\mathsf{OPair}(p) \;\wedge \\ & (\forall p_1, p_2 \in f)(\forall x \in_3 f)(x = \pi^2_1(p_1) = \pi^2_1(p_2) \;\rightarrow\; p_1 = p_2).\end{aligned}$$

Often we will write $(\forall x \in \mathsf{dom}\, f)\varphi$ in place of $(\forall x \in_3 f)(x \in \mathsf{dom}\, f \rightarrow \varphi)$ and $(\forall y \in \pi^2_1[\mathsf{dom}\, f])\varphi$ in place of $(\forall p \in f)(\forall y \in_5 f)(\forall x \in_3 f)\big((y = \pi^2_1(x) \;\wedge\; x \in \mathsf{dom}\, f) \;\rightarrow\; \varphi\big)$, when f is a function and we want to quantify over the first projection of the elements of its domain. In general, to increase clarity, these compact versions of restricted quantifiers will be used, with their meanings being explicit, even though not specified. When s is a set of n-tuples, we will write $\pi^n_i[s]$ to intend the set of the i-th projections of all elements of s, namely

$$x \in \pi^n_i[s] \quad \overset{\text{Def}}{\longleftrightarrow} \quad (\exists t \in s)\; x = \pi^n_i(t)\,.$$

We will also make use of the following function related notions, which work only under the assumption that f is a function:

$$\begin{aligned}
x \in \mathsf{dom}\, f \;&\overset{\text{Def}}{\longleftrightarrow}\; (\exists p \in f)x = \pi_1(p)\,, && (\exists)_0\\
d = \mathsf{dom}\, f \;&\overset{\text{Def}}{\longleftrightarrow}\; (\forall x \in d)(x \in \mathsf{dom}\, f) \;\wedge\; (\forall x \in \mathsf{dom}\, f)(x \in d)\,, && (\forall\exists)_0\\
x \in \mathsf{ran}\, f \;&\overset{\text{Def}}{\longleftrightarrow}\; (\exists p \in f)x = \pi^2_2(p)\,, && (\exists)_0\\
y = f(x) \;&\overset{\text{Def}}{\longleftrightarrow}\; (\exists p \in f)(x = \pi^2_1(p) \;\wedge\; y = \pi^2_2(p))\,, && (\exists)_0\\
v \in f(x) \;&\overset{\text{Def}}{\longleftrightarrow}\; (\exists y \in \mathsf{ran}\, f)(y = f(x) \;\wedge\; v \in y)\,, && (\exists)_0\\
\mathsf{Fun}(f(x)) \;&\overset{\text{Def}}{\longleftrightarrow}\; (\forall y \in \mathsf{ran}\, f)(y = f(x) \rightarrow \mathsf{Fun}(y))\,, && (\forall\exists)_0\\
v \in \mathsf{dom}\, f(x) \;&\overset{\text{Def}}{\longleftrightarrow}\; (\exists y \in \mathsf{ran}\, f)(y = f(x) \;\wedge\; v \in \mathsf{dom}\, y)\,, && (\exists)_0\\
f(x) = g(y) \;&\overset{\text{Def}}{\longleftrightarrow}\; (\forall v \in \mathsf{ran}\, f)(\forall w \in \mathsf{ran}\, f)(v = f(x) \;\wedge\; w = g(y) \;\rightarrow\; v = w). && (\forall)_0
\end{aligned}$$

Similar predicates, not listed here, will be used throughout.

Natural numbers are classically represented as finite ordinals. Whilst the definition of ordinals is $(\forall)_0$, hence it has a very low syntactic complexity (in the sense explained in the Introduction), expressing their finitude in weak theories requires more effort. We put:

$$\begin{aligned}
s = \varnothing \;&\overset{\text{Def}}{\longleftrightarrow}\; (\forall x \in s)x \neq x, && (\forall)_0\\
t = s^+ \;&\overset{\text{Def}}{\longleftrightarrow}\; (\forall x \in t)(x = s \;\vee\; x \in s), && (\forall)_0\\
t = s^- \;&\overset{\text{Def}}{\longleftrightarrow}\; (s = t^+) \;\vee\; (t = \varnothing \;\wedge\; s = \varnothing), && (\forall)_0
\end{aligned}$$

where $s^+ \coloneqq s \cup \{s\}$ and s^- are, resp., the successor and the predecessor of s. Under **(E)**, **(N)**, **(W)**, **(L)**, and **(R)**, naturals are expressed by the following $(\forall\exists\forall)_0$-specifications:

$$\begin{aligned}
\mathsf{Trans}(X) \;&\overset{\text{Def}}{\longleftrightarrow}\; (\forall\, v \in X)\,(\forall\, u \in v)\; u \in X\,,\\
\mathsf{Ord}(X) \;&\overset{\text{Def}}{\longleftrightarrow}\; \mathsf{Trans}(X) \;\wedge\; (\forall\, u, v \in X)\,\big(\,u \in v \;\vee\; v \in u \;\vee\; v = u\,\big)\,,\\
\mathsf{Num}(X) \;&\overset{\text{Def}}{\longleftrightarrow}\; (\forall\, t \in X)\,(\exists\, y \in X)\,(\forall\, u \in X)\,(\,u = y \vee u \in y\,) \;\wedge\\
&\quad (\forall\, y \in X)\,(\forall\, t \in y)\,(\exists\, z \in y)\,(\forall\, u \in y)\,(\,u = z \vee u \in z\,) \;\wedge \mathsf{Ord}(X).
\end{aligned}$$

In T_s, namely under the separation axiom schema (**Sep**), the latter can be simplified into the following simpler $(\forall\exists\forall)_0$-formula:

$$\mathsf{Num}(X) \overset{\text{Def}}{\longleftrightarrow} (\forall\, y \in X^+)\,\big(\, y = \varnothing \vee (\exists\, z \in X)\; z^+ = y\,\big) \wedge \\ (\forall\, u, v \in X\ \mathtt{less}\ \varnothing)\,(\, u \in v \vee v \in u \vee v = u\,)\,.$$

Note that in T_f, Num coincides with Ord, and therefore belongs to the class $(\forall)_0$. In other extensions of T, in order to lower the complexity of the Δ_0-specification of natural numbers, we must overload the structure of this property. We can think of natural numbers as specially constrained quadruples; specifically, we endow the classical finite ordinal with information on its predecessors and successors, allowing us to use existential formulae instead of universal ones to express them:

$$\begin{aligned}\mathsf{Num}(Q) \overset{\text{Def}}{\longleftrightarrow}\ & \mathsf{Quadruple}(Q)\ \wedge\ \mathsf{Fun}(\pi_4(Q))\ \wedge\ \pi_1(Q) = \pi_2(Q)^-\ \wedge\ \pi_3(Q) = \pi_2(Q)^+ \wedge \\ & (\forall n \in \pi_2(Q))(\mathsf{Triple}(\pi_4(Q)(n))\ \wedge\ \mathsf{dom}\,\pi_4(Q) = \pi_2(Q) \wedge \\ & (\forall t \in \mathsf{ran}\,\pi_4(Q))(\pi_1(t) = \pi_2(t)^-\ \wedge\ \pi_3(t) = \pi_2(t)^+) \wedge \\ & (\forall y \in \pi_3(Q))\big(y = \varnothing\ \vee\ (\exists z \in \pi_2(Q))\pi_3(\pi_4(Q)(z)) = y\big) \wedge \\ & (\forall u, v \in \pi_2(Q))(u \in v\ \vee\ v \in u\ \vee\ u = v).\end{aligned}$$

Here the *definiendum* requires that: (i) the second projection of a natural number is a (finite) ordinal, (ii) its first and third projections are respectively the predecessor and successor of the second projection, (iii) its fourth projection is a function whose domain is the second projection mapping (finite) ordinals n to triples t such that $\pi_2(t) = n$, and (iv) the first and third projections of t are the predecessor and the successor of n respectively. Thanks to these conditions, statable with a single quantifier alternation, infinite ordinals are ruled out.

All of the following definitions can be straightforwardly re-adapted to use this last $(\forall\exists)_0$-formula; however, to enhance readability we will continue to refer to our preceding characterizations of natural numbers in the definitions that follow. Given a numeral $\overline{n}$, we can easily express the predicate $x = \overline{n}$ by means of a Σ_1-formula having a $(\forall)_0$ matrix:

$$x = \overline{n} \overset{\text{Def}}{\longleftrightarrow} (\exists x_0)\cdots(\exists x_{n-1})\ (x_0 = \varnothing\ \wedge\ \textstyle\bigwedge_{i=1}^{n-1} x_i = x_{i-1}^+\ \wedge\ x = x_{n-1}^+).$$

The following claims are plain (cf. [10]):

Lemma 1. *In T, the following are provable:*

1. $(\overline{n})^+ = \overline{n+1}$, $\quad x = \overline{n}\ \wedge\ y = x^+\ \rightarrow\ y = \overline{n+1}$,
2. $x^+ = \overline{n+1} \rightarrow x = \overline{n}$, $\quad \mathsf{Num}(x) \rightarrow x \not< x$,
3. $\mathsf{Num}(x)\ \wedge\ y \in x\ \rightarrow\ \mathsf{Num}(y)$, $\mathsf{Num}(x)\ \wedge\ \mathsf{Num}(y)\ \wedge\ x \neq y\ \rightarrow x^+ \neq y^+$,
4. $\mathsf{Num}(x)\ \wedge \mathsf{Num}(y)\ \wedge\ y \leq x\ \rightarrow\ x = y\ \vee\ x < y$,
5. $\mathsf{Num}(\overline{0})$, $\quad \mathsf{Num}(\overline{n})$, $\mathsf{Num}(x) \rightarrow \mathsf{Num}(x^+)$, $\quad \forall x(\mathsf{Num}(x) \rightarrow x \not< \overline{0})$,
6. $\forall x(\mathsf{Num}(x) \rightarrow x < \overline{n}\ \vee\ x = \overline{n}\ \vee\ \overline{n} < x)$,
7. $\forall x\big(\mathsf{Num}(x) \rightarrow (x < \overline{n+1} \leftrightarrow x = \overline{0}\ \vee\ \cdots\ \vee\ x = \overline{n})\big)$;
8. *if $n < m$, then $T \vdash \overline{n} < \overline{m}$;* *if $n \neq m$, then $T \vdash \overline{n} \neq \overline{m}$.*

Hereafter, we place ourselves in a generic extension T' of T, such as T_s, where a formal counterpart of the concept of natural number has been cast as a $(\forall\exists)_0$-formula.

Provided that T' preserves the previous lemma (under retouched definitions), the following holds.

Theorem 1 ([10]). *Every total recursive n-ary function f on $\mathbb{N} \coloneqq \{0, 1, 2, \dots\}$ is* strongly representable *in T', in the sense that there is a formula φ such that for $k_1, \dots, k_n, k \in \mathbb{N}$:*

- $f(k_1, \dots, k_n) = k \quad implies \quad T' \vdash \varphi(\overline{k}_1, \dots, \overline{k}_n, \overline{k})$, *and*
- $T' \vdash \exists\, x\, \forall\, y \,\big(\, \varphi(\overline{k}_1, \dots, \overline{k}_n, y) \;\leftrightarrow\; y = x \,\big)$.

3 Codes

We will revamp the original definition of code in [10] in order to give it a more explicit structure. Our codes are finite-length sequences that represent the syntax trees of formulae by means of a linear structure. As such, the first element is a natural number, whose presence allows us to use restricted quantifiers, while the remaining ones are triples that encode the nodes of the tree. Each triple contains the node type in the first component and either two leaves (variable nodes) or pointers to nodes previously appearing in the sequence. The last triple in the node is considered to be the root of the tree. Clearly, for each formula φ, there is a countable infinity of code sequences encoding φ.
In addition, in order to be able to further simplify the definition, we will make use of unordered functions in our definitions.

Definition 2 (Code sequence).

$$\begin{aligned}\mathsf{Seq}_C(f) \overset{Def}{\longleftrightarrow}\ & \mathsf{Num}(f(\overline{0})) \ \wedge\ \mathsf{dom}(f) \in f(\overline{0}) \ \wedge \\ & (\forall p \in f)\big(|p| \leq 2 \ \wedge\ (\exists x, y \in p)(x \neq y)\big) \ \wedge \\ & (\forall p \in f)(\forall x, y \in p)\big[x \neq y \ \wedge\ x \neq \overline{0} \ \wedge\ y \neq \overline{0} \ \rightarrow \\ & \qquad y \notin x \ \wedge\ x \notin y \ \wedge\ \big(\mathsf{Num}(x) \ \vee\ \mathsf{Num}(y)\big)\big].\end{aligned}$$

The third condition forces f to be a sequence of doubletons proper, whilst the fourth one establishes that each pair, each element of f, save $\overline{0}$ paired with its image, is made of a number and a non-number. The first two conjuncts state that the first element (height of the sequence) is a natural number and that the domain (length) of the sequence is bounded by it. The literal $|p| \leq 2$ can plainly be expressed by $(\forall x, y, z \in p)(x = y \ \vee\ y = z \ \vee\ x = z)$. A natural specification for $y = f\restriction x$ is:

$$x \in \mathsf{dom}_C\, f \overset{\text{Def}}{\longleftrightarrow} (\exists p \in f)(\exists u \in p)[x \in p \ \wedge\ x \neq u \ \wedge\ (x \in u \ \vee\ \overline{0} \in x)],$$
$$y = f\restriction x \overset{\text{Def}}{\longleftrightarrow} x \in \mathsf{dom}_C\, f \ \wedge\ (\exists p \in f)\, x, y \in p \ \wedge\ y \neq x \ \wedge\ (x \in y \rightarrow x = \overline{0}),$$

where $x \in \mathsf{dom}_C\, f$ means "x is in the unordered domain of the code sequence f". These two definitions have the desired meanings; in fact, as the length of any code is at least 2, its height must exceed 2. Hence $\overline{0} \in f\restriction \overline{0}$ whenever $\mathsf{Seq}_C(f)$ holds. The basic principle that makes them work is that, thanks to regularity, it is always possible to prove that two naturals are comparable by membership under any definition we have considered. Combining this result with the fact that no ordered pair, and hence no triple, satisfies the predicate Num, it is always possible to distinguish the element in the range from the one in the domain.

Thence, the complexity of the formula entirely depends on the complexity of Num: the formula is $(\forall\exists)_0$ whenever Num is $(\forall\exists)_0$ and $(\forall\exists\forall)_0$ when it is $(\forall\exists\forall)_0$. We will often write f_x instead of $f\restriction x$, and we will also make use of the following specification in the coming sections:

$$y \in \mathsf{ran}_C\, f \overset{\text{Def}}{\longleftrightarrow} (\exists p \in f)(\exists x \in p)(x \in y \ \vee\ \overline{0} \notin y).$$

Definition 3 (Code).

$$\begin{aligned}\mathsf{Cod}(f) \overset{Def}{\longleftrightarrow}\ & \mathsf{Seq}_C(f) \ \wedge\ (\exists p \in f)(\exists q \in f)(p \neq q) \wedge \\ & (\forall i \in \mathsf{dom}_C\, f)[i \neq \overline{0} \ \to \mathsf{Triple}(f_i) \ \wedge\ \mathsf{Symbol}(\pi_1(f_i)) \wedge \\ & (\pi_2(f_i) \in f\restriction\overline{0} \ \vee\ \pi_2(f_i) \in i) \ \wedge (\pi_3(f_i) \in f\restriction\overline{0} \ \vee\ \pi_3(f_i) \in i)].\end{aligned}$$

We already noted that Triple is $(\forall\exists)_0$, and the last two conjuncts in the succedent of the implication are $(\exists)_0$, hence it has the same Δ_0-complexity of Seq_C. The first projection of a triple is a Symbol set, i.e., a set that represents a connective or a relator in $\mathcal{L}_\in$. In practice, Symbol can be thought as Num, as we require just a countable amount of them. The second and third projections of the triples are either indices of variables (relative to their standard ordering $\nu_0, \nu_1, \nu_2, \dots$) or pointers to previous nodes. Pointers of previous nodes precede i, whilst we impose that the index of a variable shall be less than the height of the code. Note that this restriction does not reduce the power of codes, as it is sufficient to increase the height in order to be able to encode any variable.

A total order on codes. The given definition, $\mathsf{Cod}(f)$, of code implies a natural total order on the class of codes. We put:

$$\begin{aligned}f \leq_C g \overset{\text{Def}}{\longleftrightarrow}\ & [f\restriction\overline{0} < g\restriction\overline{0}] \ \vee\ [f\restriction\overline{0} = g\restriction\overline{0} \ \wedge\ \mathsf{dom}_C\, f < \mathsf{dom}_C\, g] \vee \\ & \big[f\restriction\overline{0} = g\restriction\overline{0} \ \wedge\ \mathsf{dom}_C\, f = \mathsf{dom}_C\, g \wedge \\ & \quad (\forall m \in \mathsf{dom}_C\, f)([(\forall i \in \mathsf{dom}_C\, f)(m \in i \to f_i = g_i) \ \wedge \\ & \qquad\qquad f\restriction m \neq g\restriction m] \ \to \varphi_{\mathtt{less}}(f, g, m))\big],\end{aligned}$$

where $\varphi_{\mathtt{less}}(f, g, m)$ stands for:

$$\begin{aligned}&\big[\pi_3(f_m) <_S \pi_3(g_m)\big] \ \vee\ \big[\pi_3(f_m) = \pi_3(g_m) \ \wedge\ \pi_2(f_m) < \pi_2(g_m)\big] \vee \\ &\big[\pi_3(f_m) = \pi_3(g_m) \ \wedge\ \pi_2(f_m) = \pi_2(g_m) \ \wedge\ \pi_1(f_m) < \pi_1(g_m)\big].\end{aligned}$$

By $<_S$, we denote a strict total order on the class of symbols. Since the projections of f can be extracted with $(\exists)_0$-formulae, and $\mathsf{dom}_C\, f = \mathsf{dom}_C\, g$ can be checked with a $(\forall\exists)_0$-formula, the complexity is mostly dependent on $<_S$. Under suitable definitions of Symbol and $<_S$ (e.g., if we identify symbols with natural numbers up to some finite bound n), the formula is $(\forall\exists)_0$.
The following proposition is trivially proved by contradiction:

Lemma 2. *In T, if $<_S$ is linear, then so is $\leq_C$.*

We can also explicitly define the strict variant of $\leq_C$ with a $(\forall\exists)_0$-formula:

$$f <_C g \quad \overset{\text{Def}}{\longleftrightarrow} \quad f \leq_C g \ \wedge\ (\exists p \in g)(p \notin f).$$

We will now specify a successor function which allows us to enumerate the class of all the code sequences. To achieve simpler specifications, we will consider natural numbers below $f\restriction\overline{0}$ as symbols. We have three cases:

1. There is some triple in which one of the values is not $f\restriction\overline{0} - 1$: in this case, we interpret the sequence of all triples as a base $f\restriction\overline{0}$–number and take its successor.

2. None of the triples can be increased, but the domain of the code is smaller than its height: in this case, the successor is the code which has the same height, one more triple, and whose triples are made up of zeroes.
3. Neither of the previous cases holds: in this case the successor has the height increased by one, and has just one triple of zeroes.

This informal definition can be written as: $g = \mathsf{Next}_C(f) \overset{\text{Def}}{\longleftrightarrow} (C_1) \vee (C_2) \vee (C_3)$.

Condition (C_1): We define two utility predicates. The first one, $y = \mathsf{Next}_T(x, c, v)$, is true when y is the successor triple of x, thinking of x as a number in base c. The variable v is conveniently used in the antecedent of the implications in order to be able to write a $(\exists)_0$-formula.

$$\begin{aligned} y = \mathsf{Next}_T(x,c,v) \overset{\text{Def}}{\longleftrightarrow}\ & (\pi_1(x) \neq c \wedge v = \pi_1(x)^+ \ \rightarrow\ \pi_1(y) = v) \wedge \\ & (\pi_1(x) = c \wedge \pi_2(x) \neq c \wedge v = \pi_2(x)^+ \ \rightarrow\ \pi_2(y) = v) \wedge \\ & (\pi_1(x) = c \wedge \pi_2(x) = c \wedge \pi_3(x) \neq c \wedge v = \pi_3(x)^+ \ \rightarrow\ \pi_3(y) = v). \end{aligned}$$

The second formula, $\mathsf{Carry}_T(f, c, i)$, is true when the i-th triple of the sequence is the last one that takes the carry in the successor operation considering the sequence as a number in base c. This is a $(\forall)_0$-formula. We have:

$$\begin{aligned} \mathsf{Carry}_T(f,c,i) \overset{\text{Def}}{\longleftrightarrow}\ & (\forall j \in i)(\forall v_1 \in \pi_1(\mathsf{ran}_C\, f))(\forall v_2 \in \pi_2(\mathsf{ran}_C\, f))(\forall v_3 \in \pi_3(\mathsf{ran}_C\, f)) \\ & (\forall u_1 \in \pi_1(\mathsf{ran}_C\, f))(\forall u_2 \in \pi_2(\mathsf{ran}_C\, f))(\forall u_3 \in \pi_3(\mathsf{ran}_C\, f)) \\ & [j \neq \overline{0} \ \wedge\ v_1 = \pi_1(f_j) \ \wedge\ v_2 = \pi_2(f_j) \ \wedge\ v_3 = \pi_3(f_j) \wedge \\ & \qquad u_1 = \pi_1(f_i) \ \wedge\ u_2 = \pi_2(f_i) \ \wedge\ u_3 = \pi_3(f_i) \ \rightarrow \\ & \qquad v_1 = c \ \wedge v_2 = c \ \wedge v_3 = c \ \wedge\ (u_1 \neq c \ \vee\ u_2 \neq c \ \vee\ u_3 \neq c)]. \end{aligned}$$

Now we are ready to produce a compact $(\forall\exists)_0$-specification of Condition (C_1).

$$\begin{aligned} & (\exists m \in \mathsf{dom}_C\, f)(\exists y \in f\restriction\overline{0})[m \neq \overline{0} \ \wedge\ (\pi_1(f_m) \in y \ \vee\ \pi_2(f_m) \in y \ \vee\ \pi_3(f_m) \in y)] \wedge \\ & (\forall m \in \mathsf{dom}_C\, f)(\forall c \in f\restriction\overline{0})(\forall i \in m)[f\restriction\overline{0} = c^+ \ \wedge\ \mathsf{Carry}_T(f,c,m) \wedge \\ & \qquad\qquad (\forall j \in m)\mathsf{Carry}_T(f,c,j) \rightarrow f_i = (0,0,0)] \wedge \\ & (\forall m \in \mathsf{dom}_C\, f)(\forall c \in f\restriction\overline{0})(\forall v_1 \in \pi_1(\mathsf{ran}_C\, g))(\forall v_2 \in \pi_2(\mathsf{ran}_C\, g))(\forall v_3 \in \pi_3(\mathsf{ran}_C\, g))) \\ & [f\restriction\overline{0} = c^+ \ \wedge\ \mathsf{Carry}_T(f,c,m) \rightarrow g_m = \mathsf{Next}_T(f_m,c,v_1) \ \vee\ g_m = \mathsf{Next}_T(f_m,c,v_2) \vee \\ & \qquad\qquad g_m = \mathsf{Next}_T(f_m,c,v_3)], \end{aligned}$$

where $f_i = (0,0,0)$ is syntactic sugar for $\pi_1(f_i) = \overline{0} \ \wedge\ \pi_2(f_i) = \overline{0} \ \wedge\ \pi_3(f_i) = \overline{0}$.

Condition (C_2): (C_2) is a $(\forall\exists)_0$-formula:

$$\begin{aligned} & (\forall x \in \mathsf{dom}_C\, f)(\forall y \in f\restriction\overline{0})(f\restriction\overline{0} = y^+ \ \wedge\ x \neq \overline{0} \rightarrow f_x = (y,y,y)) \wedge \\ & \mathsf{dom}_C\, f < f\restriction\overline{0} \ \wedge\ g\restriction\overline{0} = f\restriction\overline{0} \ \wedge\ (\mathsf{dom}_C\, f)^+ = \mathsf{dom}_C\, g \wedge \\ & (\forall i \in \mathsf{dom}_C\, g)(i \neq \overline{0} \rightarrow g_i = (0,0,0)). \end{aligned}$$

Condition (C_3): With this last $(\forall\exists)_0$-condition, we cover every possible case:

$$\begin{aligned} & (\forall x \in \mathsf{dom}_C\, f)(\forall y \in f\restriction\overline{0})(f\restriction\overline{0} = y^+ \ \wedge\ x \neq \overline{0} \rightarrow f_x = (y,y,y)) \wedge \\ & \mathsf{dom}_C\, f = f\restriction\overline{0} \ \wedge\ g\restriction\overline{0} = f\restriction\overline{0}^+ \ \wedge\ \mathsf{dom}_C\, g = \overline{2} \ \wedge\ g(\overline{1}) = (\overline{0},\overline{0},\overline{0}). \end{aligned}$$

The previous discussion readily yields

Lemma 3. *$\mathsf{Next}_C(f)$ is a $(\forall\exists)_0$-formula.*

It is clear that starting from the code $\{\{0,3\},\{1,\langle 0,0,0\rangle\}\}$, through successive application of Next_C, we can enumerate all the codes, as, given a length and a height, there is a finite amount of codes with that length and height. The bottom code, $\overline{0}_C$, is characterized by the $(\forall\exists)_0$-formula

$$\begin{aligned} f = \overline{0}_C \overset{\text{Def}}{\longleftrightarrow}\ & \mathsf{Cod}(f) \ \wedge\ f_0 = \overline{2} \ \wedge\ \mathsf{Triple}(f_1) \wedge \\ & \pi_1(f_1) = \overline{0} \ \wedge\ \pi_2(f_1) = \overline{0} \ \wedge\ \pi_3(f_1) = \overline{0}. \end{aligned}$$

Throughout the rest of the section, we will state some useful facts about codes that will be essential in developing an argument *à la* Gödel within the weak set theories we are considering. As most of these facts admit proofs very similar to the ones available in [10], we will not provide details.

Lemma 4. *In T:*
(a) $\forall x \forall y \big(\mathsf{Cod}(x) \ \wedge \ \mathsf{Next}_C(x) = y \ \rightarrow \ \mathsf{Cod}(y)\big)$;
(b) $\forall x \forall y \forall z \big(\mathsf{Cod}(x) \ \wedge \ \mathsf{Cod}(y) \ \wedge \ \mathsf{Cod}(z) \ \wedge \ x \leq_C z \ \wedge \ z \leq_C y \ \wedge \ y = \mathsf{Next}_C(x) \ \rightarrow \ z = x \ \vee \ z = y\big)$;
(c) $\forall x \forall y \big(\mathsf{Cod}(x) \ \wedge \ \mathsf{Cod}(y) \ \wedge \ \mathsf{Next}_C(x) = y \ \rightarrow \ x <_C y\big)$;
(d) $\forall x \forall y \forall z \big(\mathsf{Cod}(y) \ \wedge \ \mathsf{Next}_C(y) = x \ \wedge \ \mathsf{Next}_C(y) = z \ \rightarrow \ x = z\big)$.

For every natural number k, by $x = \overline{(k)}_C$ we mean:

$$(\exists x_0, \ldots, x_{k-1}) \Big(x_0 = \overline{0}_C \ \wedge \ \textstyle\bigwedge_{i=0}^{k-1} x_{i+1} = \mathsf{Next}_C(x_i) \ \wedge \ x = x_k \Big).$$

Lemma 5. *For each natural number k, we have in T:*
(a) $\overline{(k+1)}_C = \mathsf{Next}_C(\overline{(k)}_C)$; *(b)* $x = \overline{(k)}_C \ \wedge \ y = \mathsf{Next}_C(x) \ \rightarrow = \overline{(k)}_C$;
(c) $\forall x (\mathsf{Cod}(x) \ \wedge \ \overline{(k+1)}_C = \mathsf{Next}_C(x) \ \rightarrow \ x = \overline{(k)}_C$; *(d)* $\mathsf{Cod}(\overline{(k)}_C)$;
(e) $\forall x \big(\mathsf{Cod}(x) \ \rightarrow \ \neg <_C \overline{0}_C\big)$; *(f)* $\forall x \big(\mathsf{Cod}(x) \ \rightarrow \ (x \leq_C \overline{(k)}_C \leftrightarrow \ x <_C \overline{(k+1)}_C)\big)$;
(g) $\forall x \big(\mathsf{Cod}(x) \ \rightarrow \ (x <_C \overline{(k+1)}_C \ \leftrightarrow \ x = \overline{0}_C \ \vee \ \ldots \ \vee \ x = \overline{(k)}_C)\big)$;
(h) $\forall x \big(\mathsf{Cod}(x) \ \rightarrow \ (x <_C \overline{(k)}_C \ \vee \ x = \overline{(k)}_C \ \vee \ \overline{(k)}_C <_C x)\big)$.

Corollary 1. *For all natural numbers h and k, we have that if $h = k$, then $T \vdash \overline{(h)}_C = \overline{(k)}_C$; if $h < k$, then $T \vdash \overline{(h)}_C <_C \overline{(k)}_C$.*

4 Formulae

We require the following symbols (and the related identifying predicates): $\mathsf{Symbol}_{\Rightarrow}$, $\mathsf{Symbol}_{\forall}$, $\mathsf{Symbol}_{\in}$, $\mathsf{Symbol}_{=}$, and a renaming symbol for each variable v_i, Symbol_{R_i}. The intent of the last predicate is to integrate a rename operator for each variable in the language. Each one of the symbols has two parameters, either variables or subformulae, thus we add predicates to recognize the type (the symbol on top of the syntax tree of the formula) of code sequences. Remember that the topmost node of the generation tree is the last element of a code sequence. We start by defining predicates that recognize the function of a triple t of index i inside a code f of height c:

$$\begin{aligned}
\mathsf{impl}(f,t,i,c) &\overset{\text{Def}}{\longleftrightarrow} \mathsf{Symbol}_{\Rightarrow}(\pi_1(t)) \ \wedge \ \pi_2(t) \in i \ \wedge \pi_2(t) \neq \overline{0} \ \wedge \ \pi_3(t) \in i \ \wedge \ \pi_3(t) \neq \overline{0},\\
\mathsf{forall}(f,t,i,c) &\overset{\text{Def}}{\longleftrightarrow} \mathsf{Symbol}_{\forall}(\pi_1(t)) \ \wedge \ \pi_2(t) < c \ \wedge \ \pi_3(t) < i \ \wedge \ \pi_3(t) \neq \overline{0},\\
\mathsf{rename}(f,t,i,c) &\overset{\text{Def}}{\longleftrightarrow} \mathsf{Symbol}_{R}(\pi_1(t)) \ \wedge \ \pi_2(t) < c \ \wedge \ \pi_3(t) < i \ \wedge \ \pi_3(t) \neq \overline{0},\\
\mathsf{equals}(f,t,i,c) &\overset{\text{Def}}{\longleftrightarrow} \mathsf{Symbol}_{=}(\pi_1(t)) \ \wedge \ \pi_2(t) < c \ \wedge \ \pi_3(t) < c.\\
\mathsf{in}(f,t,i,c) &\overset{\text{Def}}{\longleftrightarrow} \mathsf{Symbol}_{\in}(\pi_1(t)) \ \wedge \ \pi_2(t) < c \ \wedge \ \pi_3(t) < c.
\end{aligned}$$

All of these formulae are $(\exists)_0$. We can now define the formula predicate, which holds when every node of some code is one of the elementary nodes.

Definition 4 (Code of a formula).

$$\begin{aligned}
\mathsf{Form}(f) \overset{Def}{\longleftrightarrow} (\forall c \in \mathsf{ran}_C f)(\forall i \in \mathsf{dom}_C f)(\forall t \in \mathsf{ran}_C f)[c = f \restriction \overline{0} \ \wedge \ t = f_i \ \rightarrow \\
\mathsf{impl}(f,t,i,c) \ \vee \mathsf{forall}(f,t,i,c) \ \vee \ \mathsf{rename}(f,t,i,c) \ \vee \\
\mathsf{equals}(f,t,i,c) \ \vee \ \mathsf{in}(f,t,i,c)] \ \wedge \ \mathsf{Cod}(f).
\end{aligned}$$

The formula is clearly $(\forall\exists)_0$. We can also straightforwardly define predicate symbols for derived connectives and quantifiers as we can express $\bot$ as $\forall\, v_0(v_0 \in v_0)$ and thus $\neg\varphi$ as $\varphi \to \bot$; accordingly, $\exists v\varphi$ will stand for $\big(\forall v(\varphi \to \bot)\big) \to \bot$. Consider now the following axiomatization of first-order logic with axioms:

(A1) $(((((\varphi \to \psi) \to ((\chi \to \bot) \to (\theta \to \bot))) \to \chi) \to \tau) \to ((\tau \to \varphi) \to (\theta \to \varphi)))$;
(A2) $\forall v_i\big((\varphi \to \psi) \to (\varphi \to \forall v_i(\psi))\big)$ (v_i not free in φ);
(A3) $\forall v_i(\varphi(v_i) \to \varphi(v_j))$;
(A4) $\big(\forall v_i\big((\varphi \to (\forall v_i\varphi \to \bot)) \to \bot\big)\big) \to \bot$;
(A5) $x = x$;
(A6) $x = y \to (\varphi(x) \to \varphi(y))$.

For each such schema, we can write a $(\forall\exists)_0$-formula that recognizes whether the code of a formula is in the schema. As the reader can check, most of these specifications are rather trivial. However, the specifications of (A2), (A3), and (A6) are more problematic. Formulae that are instances of the schema (A3) are captured by the following predicate $\mathsf{A3}$:

$$\begin{aligned}\mathsf{A3}(f) \overset{\text{Def}}{\longleftrightarrow}\ & (\forall x \in \mathsf{dom}_C\, f)(\forall u \in_4 f)(\forall v \in_5 f)(\forall i \in_4 f)(\forall t \in_5 f)\\ & \big[x^+ = \mathsf{dom}_C\, f \ \wedge \pi_2(f_x) = u \ \wedge \pi_3(f_x) = v \ \wedge\ \pi_2(f_u) = i \ \wedge \pi_3(f_u) = t \ \to\\ & \qquad \mathsf{impl}(f, f_x, x, f\restriction\overline{0}) \ \wedge\ \mathsf{forall}(f, f_u, u, f\restriction\overline{0}) \ \wedge\\ & \qquad (\exists j \in f\restriction\overline{0})[\mathsf{rename}_i(f, f_v, v, f\restriction\overline{0}) \ \wedge\ t = \pi_3(f_i)]\big] \ \wedge\ \mathsf{Cod}(f).\end{aligned}$$

One can proceed similarly with (A6). As for (A2), we also need means to express whether a variable has free occurrences in a formula. While this problem is particularly hard to solve in general, it will be easy in the context in which it will be needed.

We are now able to define

$$\mathsf{LAxiom}(f) \quad \overset{\text{Def}}{\longleftrightarrow} \quad \mathsf{A1}(f) \vee \mathsf{A2}(f) \vee \mathsf{A3}(f) \vee \mathsf{A4}(f) \vee \mathsf{A5}(f) \vee \mathsf{A6}(f),$$

which recognizes whether the code of a formula is a first-order logic axiom. Depending on the theory we are considering, we also have several theory axioms that are usually trivial to express. Using similar specifications, one can define the TAxiom that holds whenever a formula code is an axiom in the theory. We also put:

$$\mathsf{Axiom}(f) \quad \overset{\text{Def}}{\longleftrightarrow} \quad \mathsf{LAxiom}(f) \ \vee\ \mathsf{TAxiom}(f).$$

We define next the two predicates that hold when f is a left or a right copy of g:

$$\begin{aligned}\mathsf{CLCopy}(f,g) \overset{\text{Def}}{\longleftrightarrow}\ & (\forall i \in \mathsf{dom} f)(\forall x \in \pi_2(\mathsf{ran} f))[x = \pi_2(\mathsf{last} f) \ \to\ (f_i \in \mathsf{ran}\, g \leftrightarrow\\ & \qquad (\exists j \in \mathsf{dom} f)(j \le x \ \wedge \mathsf{PtBy}(i, f_j)) \ \vee\ i = x)],\\ \mathsf{CRCopy}(f,g) \overset{\text{Def}}{\longleftrightarrow}\ & (\forall i \in \mathsf{dom} f)(\forall x \in \pi_2(\mathsf{ran} f))[x = \pi_3(\mathsf{last} f) \ \to\ (f_i \in \mathsf{ran}\, g \leftrightarrow\\ & \qquad (\exists j \in \mathsf{dom} f)(j \le x \ \wedge \mathsf{PtBy}(i, f_j)) \ \vee\ i = x)].\end{aligned}$$

The formula $\mathsf{PtBy}(i,t)$ holds when the triple t has a pointer to the ith node. This is clearly an $(\exists)_0$-formula, which can be written as:

$$\begin{aligned}\mathsf{PtBy}(i,t) \overset{\text{Def}}{\longleftrightarrow}\ & \big(\mathsf{Symbol}_{\Rightarrow}(\pi_1(t)) \ \wedge\ (i = \pi_2(t) \ \vee\ i = \pi_3(t))\big) \vee\\ & (\mathsf{Symbol}_{\forall}(\pi_1(t)) \ \wedge\ i = \pi_3(t)) \vee\\ & (\mathsf{Symbol}_{R}(\pi_1(t)) \ \wedge\ i = \pi_3(t)).\end{aligned}$$

The temporary notation $\mathsf{last}\, f$ can be eliminated by means of the rewriting rules explained below:

- add $(\forall n \in \mathsf{dom}\ f)$, involving a new n, in front of the quantificational prefix;
- conjoin $n^+ = \mathsf{dom}\ f$ with the antecedent of the matrix;

– replace every occurrence of $\mathsf{last}\ f$ by the term f_n.

The added complexity depends on the complexity of checking $n^+ = \mathsf{dom}\ f$ that normally has a $\forall\exists$-prefix. Although this would yield a $(\forall\exists\forall)_0$-formula, we will use it in a context in which we will be able to rewrite it as an $(\exists\forall)_0$-formula, hence we will be able to write the two predicates with $(\forall\exists)_0$-formulae.
Given a code c, in order to recognize bound variables, we will consider a sequence of the same length that contains the bound variables in each triple (subformula). The following $(\forall\exists)_0$-predicate establishes that b is the bound list of a code c:

$$\begin{aligned}\mathsf{BoundList}(b,c) \overset{\text{Def}}{\longleftrightarrow}\ & \mathsf{Fun}(b)\ \wedge\ \mathsf{dom}\ b = \mathsf{dom}_C\ c\ \wedge \\ & (\forall i \in \mathsf{dom}_C\ c)(\forall j \in \mathsf{dom}_C\ c)(\mathsf{PtBy}(i,b_j) \rightarrow b_j \subseteq b_i)\ \wedge \\ & (\forall i \in \mathsf{dom}_C\ c)(\forall v \in_4 b)\Big(v \in b_i\ \leftrightarrow \\ & \qquad (\exists j \in \mathsf{dom}_C\ c)(\mathsf{PtBy}(i,b_j)\ \wedge\ v \in b_j)\ \vee \\ & \qquad (\mathsf{Symbol}_\forall(\pi_1(f_i))\ \wedge\ \pi_2(f_i) = v)\Big),\end{aligned}$$

where $b_i \subseteq b_j$ is a shorthand for $(\forall x \in b_i)x \in b_j$, so that $\mathsf{BoundList}(b,c)$ is clearly $(\forall\exists)_0$. When $\mathsf{BoundList}(b,c)$ holds and we encounter a variable v in some triple c_i, it is sufficient to check if $v \in b_i$ holds in order to know whether v is bound.

Proofs. Accessing the domain and the predecessor of a natural number can be done with a $(\forall\exists)_0$- and a $(\forall)_0$-formula, respectively. This can be problematic when the former is in the antecedent of an implication and when the latter is in the succedent of some implication, as we would almost certainly end up with $\forall\exists\forall$-formulae in both cases. To solve this problem, we will embed two sequences that contain all the needed predecessors and domains in the definition of the Proof predicate.
The idea is that a proof is a sequence composed of three parts: (i) a value at index 0 that contains the point in the sequence that separates the other two parts; (ii) a list of triples, one for each one of the subformulae of the formulae that occur in the proof, that contain the subformula in the first projection, a copy of the left child in the second, and a copy of the right child in the third (between index 1 and $f\restriction\overline{0}$ - 1); and (iii) a list of indices between 1 and $f\restriction\overline{0}$ that point to the formulae that are supposed to be the steps of the proof. We also fill a list of bound variables for each of the subformulae in order to be able to always tell which variables are bound in a given formula.

$$\begin{aligned}&\mathsf{Proof}'(f,x,p,d,d_f,b) \overset{\text{Def}}{\longleftrightarrow} \\ &\quad \mathsf{Fun}(f)\ \wedge\ \mathsf{Num}(\mathsf{dom} f)\ \wedge\ \mathsf{Num}(f(\overline{0}))\ \wedge\ f(\overline{0}) \in \mathsf{dom} f\ \wedge\ d_f = \mathsf{dom} f \\ &\quad \mathsf{Fun}(p)\ \wedge\ \mathsf{dom}\ p = \mathsf{dom} f\ \wedge\ (\forall i \in \mathsf{dom}\ p)(i \neq \overline{0} \rightarrow p_i = i^-)\ \wedge \\ &\quad \mathsf{Fun}(d)\ \wedge \mathsf{dom}\ d = \mathsf{dom} f\ \wedge\ (\forall i \in \mathsf{dom}\ d)(f(\overline{0}) \in i \rightarrow d_i = \mathsf{dom}\ f_i)\ \wedge \\ &\quad (\forall i \in f(\overline{0}))[\mathsf{Triple}(f_i)\ \wedge\ \mathsf{Form}(\pi_1(f_i))\ \wedge\ \mathsf{Form}(\pi_2(f_i))\ \wedge\ \mathsf{Form}(\pi_3(f_i))\ \wedge \\ &\qquad (\mathsf{NotAtom}(f_i) \rightarrow (\exists j_1, j_2 \in i)(\pi_1(f_{j_1}) = \pi_2(f_i)\ \wedge\ \pi_1(f_{j_2}) = \pi_3(f_i)))] \wedge \\ &\quad (\forall i \in f(\overline{0}))[i \in f(\overline{0})\ \wedge\ \mathsf{NotAtom}(\pi_1(f_i))\ \wedge i \neq \overline{0} \rightarrow \\ &\qquad\qquad \mathsf{CLCopy}(\pi_1(f_i),\pi_2(f_i))\ \wedge\ \mathsf{CRCopy}(\pi_1(f_i),\pi_3(f_i))] \wedge \\ &\quad (\forall i \in \mathsf{dom} f)(f(\overline{0}) \leq i\ \rightarrow\ f_i \in f(\overline{0})\ \wedge\ f_i \neq \overline{0})\ \wedge \\ &\quad \mathsf{Fun}(b)\ \wedge\ \mathsf{dom}\ b = f(\overline{0})\ \wedge\ (\forall i \in f(\overline{0}))(\mathsf{BoundList}(b_i,\pi_1(f_i)))\ \wedge \\ &\quad (\forall i,n,j \in \mathsf{dom} f)[d_f = n^+\ \wedge\ f(\overline{0}) \leq i\ \rightarrow\ \pi_1(f(f(n)) = x\ \wedge\ \Psi],\end{aligned}$$

where the formula Ψ will be defined below. Clearly, $\mathsf{NotAtom}$ is easily definable starting from last and the Symbol predicates. This is the point in the formula in which d is implicitly used in order to be able to express last as an $(\exists)_0$-formula occurring in antecedents. All the conjuncts, but the last one, enforce that the three stated conditions hold on the formula. The last one states that x must be the conclusion of the proof steps, and with the formula Ψ defined below we intend to verify that all the steps are either axioms or results of the application of some inference rule. Given a $(\forall\exists)_0$-definition of Ψ, the whole formula plainly becomes $(\forall\exists)_0$.
The formula Ψ is the disjunction of the following four formulae:

$\mathsf{Axiom}(f_i)$. Clearly, to recognize axiom (A6), we have to use the bound variables list properly. This clearly does not raise the Δ_0-complexity, as it is sufficient to access the list that comes with just existential quantifiers.

Modus Ponens:
$$(\exists j_1, j_2 \in i)(f(\overline{0}) \ \wedge \ f(\overline{0}) \leq j_2 \ \wedge \ \mathsf{Symbol}_{\Rightarrow}(\pi_1(\mathsf{last}(\pi_1 f(f_{j_1}))) \wedge \pi_1(f(f_i)) = \pi_3(f(f_{j_1})) \ \wedge \ \pi_1(f(f_{j_2})) = \pi_2(f(f_{j_1}))).$$

Universal Generalization:
$$(\exists j_1 \in i)[f(\overline{0}) \ \wedge \ \pi_1(f(f_{j_1})) = \pi_3(f(f_i)) \ \wedge \ \mathsf{Symbol}_{\forall}(\pi_1(\mathsf{last}(\pi_1(f(f_i)))].$$

Rename Resolution:
$$(\forall k \in \mathsf{dom}\pi_3(f(f_{p_i})))(\forall t \in \mathsf{ran}\pi_3(f(f_{p_i})))(\forall j \in f(\overline{0})) \big[j = \mathsf{rfrom}(\mathsf{last} f(f_i)) \ \wedge \ t = \pi_3(f(f_{p_i}))(k) \to \psi\big] \ \wedge \ f(\overline{0}) \leq p_i,$$
where rfrom extracts the variable that has to be renamed from a rename node (depending on the encoding, with natural numbers it is $(\forall)_0$), and ψ checks is a (big) formula that forces the nodes in the formula to be a renamed version of the preceding formula. Hence, ψ just checks for each triple t if it contains the variable that has to be renamed, and states that it is renamed in $f(f_i)$. Clearly the entire formula is $(\forall\exists)_0$.

Thus Ψ and also Proof' are $(\forall\exists)_0$. Finally, we have the $(\forall\exists)_0$-specification:
$$\mathsf{Proof}(p, x) \quad \overset{\text{Def}}{\longleftrightarrow} \quad \mathsf{Quintuple}(p) \ \wedge \ \mathsf{Proof}'\big(\pi_1(p), x, \pi_2(p), \pi_3(p), \pi_4(p), \pi_5(p)\big).$$

5 Essential undecidability

What follows presupposes that an essential preliminary step for an *arithmetization of the syntax*, namely a map $\varphi \mapsto \lceil\varphi\rceil$ sending every formula into a natural number whence one can retrieve a formula φ' such that $\varphi' \dashv\vdash \varphi$, has been implemented (cf. [7]). Our next lemma will be useful in exploiting such a "Gödel numbering".

Lemma 6. *The function that associates a* Cod*, c, with its index k is strongly representable in* T' *through the existential closure of a* $(\forall\exists)_0$*-formula* $\varphi_{\mathtt{ind}}(c, k)$.

Lemma 7. *The predicate* $\mathsf{Subst}(c, t, d)$ *that holds when the formula with code d results from the formula with code c through substitution of the lowest-index variable by the code of a term t is strongly represented in* T' *by a* Σ_1*-formula with a* $(\forall\exists)_0$*-matrix* $\varphi_{\mathtt{sub}}(c, t, d)$*. More specifically, we have that for each formula* φ *and term code t:*
$$T' \vdash \forall d(\ \mathsf{Subst}(\lceil\varphi\rceil, t, d) \leftrightarrow d = \lceil\forall x(x = t \to \varphi)\rceil\ ).$$

In practice, we will write $\varphi_{\mathtt{sub}}(c,t,d,w)$, where w is an n-tuple containing all the existentially quantified variables, to intend the same formula after dropping its external existential quantifiers.

Lemma 8 (Fixpoint). *Given a formula $\varphi(c)$, there is a formula χ that has the same free variables as φ, save c, such that:*

$$T' \vdash \chi \leftrightarrow \forall c\big(c = \ulcorner\chi\urcorner \rightarrow \varphi(c)\big),$$

where $\ulcorner\chi\urcorner$ is some code for the formula χ.

Proof. Let $D(c,d,w) \equiv \varphi_{\mathtt{sub}}(c,c,d,w)$ and assume, without loss of generality, that w and d are not free in φ. By the previous lemma, we have that for every formula ψ, $\forall c\big(c = \ulcorner\psi\urcorner \rightarrow \exists d, w D(c,d,w)\big)$, and $\forall c\big(c = \ulcorner\psi\urcorner \rightarrow \forall d, w\big(D(c,d,w) \rightarrow d = \ulcorner\forall c(c = \ulcorner\psi\urcorner \rightarrow \varphi)\urcorner\big)\big)$. Putting $\psi = \forall d, w(D(c,d,w) \rightarrow \varphi(d))$, we have: $\forall c\big(c = \ulcorner\forall d, w(D(c,d,w) \rightarrow \varphi(d))\urcorner \rightarrow \forall d, w\big(D(c,d,w) \rightarrow d = \ulcorner\forall c(c = \ulcorner\forall d, w(D(c,d,w) \rightarrow \varphi(d))\urcorner \rightarrow \varphi)\urcorner\big)\big)$. Let $\chi = \forall c(c = \ulcorner\forall d, w(D(c,d,w) \rightarrow \varphi(d))\urcorner \rightarrow \forall d, w(D(c,d,w) \rightarrow \varphi(d)))$. Then: $\forall c \forall d \forall w\big(c = \ulcorner\forall d, w(D(c,d,w) \rightarrow \varphi(d))\urcorner \wedge D(c,d,w) \rightarrow d = \ulcorner\chi\urcorner\big)$. Since $\forall c\big(c = \ulcorner\psi\urcorner \rightarrow \exists d, w D(c,d,w)\big)$, we have $\exists d, w\big(c = \ulcorner\forall d, w(D(c,d,w) \rightarrow \varphi(d))\urcorner \wedge D(c,d,w)\big)$ for any c such that $c = \ulcorner\psi\urcorner$. Thus, as c, d, w are not free in φ, we have:

$$\chi \equiv \forall c\Big(c = \ulcorner\forall d, w(D(c,d,w) \rightarrow \varphi(d))\urcorner \rightarrow \forall d, w(D(c,d,w) \rightarrow \varphi(d))\Big) \leftrightarrow \varphi(\ulcorner\chi\urcorner).$$

In general, $c = \overline{(k)}_C$ is a Σ_1-formula with a $(\forall\exists)_0$-matrix, whilst χ can be seen as the universal closure of the formula obtained from φ by eliminating its unbounded quantifiers. In general, $c = \overline{(k)}_C$ is a Σ_1-formula with a $(\forall\exists)_0$-matrix, whilst χ can be seen as the universal closure of the formula obtained from φ by eliminating its unbounded quantifiers. When we take $\varphi(x) \equiv \forall p(\neg\mathsf{Proof}(p,x))$, which is a Π_1-formula with a $(\exists\forall)_0$ matrix, then also χ is a Π_1-formula with $(\exists\forall)_0$-matrix. This fact can be exploited to apply a Gödel incompleteness theorem-like argument.

Theorem 2. *If T' is Cod-consistent, i.e.,*

$$(\nexists \alpha \in L_\in)\Big(T \vdash \exists x\big(\mathsf{Cod}(x) \wedge \alpha\big) \wedge (\forall c \in \mathsf{Cod})\big(T \vdash \neg\alpha(c)\big)\Big),$$

then it is incomplete with respect to the class of Σ_1-formulae with a $(\forall\exists)_0$-matrix.

Proof. By applying the fixpoint lemma on the formula $\varphi(x) = \forall p(\neg\mathsf{Proof}(p,x))$, we obtain a formula χ such that:

$$T' \vdash \chi \leftrightarrow \forall p(\neg\mathsf{Proof}(p, \ulcorner\chi\urcorner)).$$

As T' is *Cod*-consistent, it neither proves nor disproves χ. Therefore, since $\neg\chi$ is a Σ_1-formula with a $(\forall\exists)_0$-matrix, then T' is incomplete with respect to this class of formulae. □

The usual computability notions on functions over natural numbers can naturally be extended on codes thanks to the fact that their index can be computed. If we take a recursively axiomatizable theory Θ on $\mathcal{L}_\in$, the collection of the indices of its axioms is a recursive set. Therefore, as computable functions are strongly

representable, also the collection of code indices C' is faithfully representable in T', i.e., there is a formula $\varphi_{C'}$ such that $n \in C' \Leftrightarrow T' \vdash \varphi_{C'}(\overline{n})$. We saw that also the correspondence between codes and their indices is strongly representable through the formula $\varphi_{\mathtt{ind}}$. Thus, the collection C of codes of the axioms of the theory is faithfully representable through the following formula $\varphi_C(x) \equiv \exists y(\varphi_{\mathtt{ind}}(x,y) \ \wedge \ \varphi_{C'}(y))$. Clearly, for every formula φ of $\mathcal{L}_{\in}$, assuming Cod-consistency, we have: $T' \vdash \varphi \ \Leftrightarrow \ T' \vdash \exists y\, \mathsf{Proof}(\varphi, y)$, and, in particular, when we instantiate φ with $\varphi_C(\overline{c})$: $T' \vdash \varphi_C(\overline{c}) \ \Leftrightarrow \ T' \vdash \exists y\, \mathsf{Proof}(\varphi_C(\overline{c}), y)$. Thence:

$$\mathbf{C}(x) = \exists z, u, y, w\big(z = \lceil \varphi_C \rceil \ \wedge \ \varphi_{\mathtt{sub}}(z, x, u, w) \ \wedge \ \mathsf{Proof}(u, y)\big)$$

is Σ_1-formula with a $(\forall\exists)_0$-matrix that faithfully represents the collection of codes C.
From the previous discussion, we have the following result:

Lemma 9. *If the theory T' is Cod-consistent, every r.e. collection of codes C is faithfully representable through a Σ_1-formula $\mathbf{C}$ with a $(\forall\exists)_0$-matrix.*

Hence, we can state our main result:

Theorem 3. *Let Θ be a recursively axiomatizable, Cod-consistent extension of the theory T'. Then Θ is incomplete with respect to the collection of Σ_1-formulae with a $(\forall\exists)_0$-matrix.*

Proof. Let C be the r.e. collection of codes of the theorems in Θ. It suffices to refer the fixpoint lemma to $\varphi(c) \equiv \neg\mathbf{C}(c)$. □

To resume the discussion on decidability in the Introduction, we state:

Corollary 2. *In any recursively axiomatizable, Cod-consistent extensions of either T_s or T_f, the decision problem for the collection of $(\forall\exists)_0$-formulae is algorithmically unsolvable.*

6 Conclusions

The claim just made is closely akin to a result in [1, Sec. 2], but the framework in which this paper has cast Corollary 2 is much broader. Instead of referring the undecidability of $(\forall\exists)_0$-formulae to a full-fledged set theory, we have been working under very weak, explicit axiomatic assumptions; moreover, our limiting results contribute to a general investigation on *essential* (set-theoretic) undecidability.

We have striven, while revisiting the material of [10], to balance transparency of the encodings with complexity of the undecidable collection of formulae; and yet, we expect that further work along the lines of this paper—possibly calling into play also the milestone result [8]—can improve this tradeoff.

References

1. Cantone, D., Omodeo, E.G., Panettiere, M.: From Hilbert's 10^{th} problem to slim, undecidable fragments of set theory. In: Cordasco, G., Gargano, L., Rescigno, A.A. (eds.) Proc. of the 21st Italian Conf. on Theoretical Computer Science, ICTCS 2020. CEUR Workshop Proc., vol. 2756, pp. 47–60. CEUR-WS.org (2020)
2. Costantini, S., Pitoni, V.: Towards a logic of "Inferable" for self-aware transparent logical agents. In: Musto, C., Magazzeni, D., Ruggieri, S., Semeraro, G. (eds.) Proc. of Italian Workshop on Explainable Artificial Intelligence, XAI.it@AIxIA 2020, Online Event, November 25-26, 2020. CEUR Workshop Proc., vol. 2742, pp. 68–79. CEUR-WS.org (2020)
3. Formisano, A., Omodeo, E.G., Policriti, A.: Three-variable statements of set-pairing. Theoretical Computer Science **322**(1), 147–173 (2004)
4. Hill, P.M., Lloyd, J.W.: The Gödel programming language. MIT Press (1994)
5. Jech, T.: Set Theory. Springer Monographs in Mathematics, Springer-Verlag Berlin Heidelberg, Third Millennium edn. (2003)
6. Levy, A.: A hierarchy of formulas in set theory, vol. 57. Providence, RI: American Mathematical Society (AMS) (1965)
7. Panettiere, M.: Essential undecidability: Foundations versus Proof Technology. Master's thesis, Università degli Studi di Catania, Italy, 83 pp. (July 28, 2021)
8. Parlamento, F., Policriti, A.: The logically simplest form of the infinity axiom. Proc. of the American Mathematical Society **103**(1), 274–276 (1988)
9. Parlamento, F., Policriti, A.: Undecidability results for restricted universally quantified formulae of set theory. Comm. on Pure and Applied Mathematics **46**(1), 57–73 (1993)
10. Parlamento, F., Policriti, A.: Decision procedures for elementary sublanguages of set theory. IX. Unsolvability of the decision problem for a restricted subclass of the Δ_0-formulas in set theory. Comm. on Pure and Applied Mathematics **41**(2), 221–251 (1988)
11. Robinson, R.M.: An essentially undecidable axiom system. In: Proc. of the International Congress of Mathematicians (Harvard University, Cambridge, MA, August 30–September 6, 1950). vol. 1, pp. 729–730. AMS, Providence, RI (1952)
12. Tarski, A.: Sur les ensembles fini. Fundamenta Mathematicae **VI**, 45–95 (1924)
13. Vaught, R.L.: On a theorem of Cobham concerning undecidable theories. In: Nagel, E., Suppes, P., Tarski, A. (eds.) Proc. of the 1960 International Congress on Logic, Methodology, and Philosophy of Science. pp. 14–25. Stanford Univ. Press (1962)