Reconciling transparency, low Δ_0-complexity and axiomatic weakness in undecidability proofs

DOMENICO CANTONE, *Department of Mathematics and Computer Science, University of Catania, Italy.*

EUGENIO G. OMODEO, *Department of Mathematics and Earth Sciences, University of Trieste, Italy.*

MATTIA PANETTIERE, *Department of Ethics, Governance and Society, Vrije Universiteit Amsterdam, The Netherlands.*

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 tailor arguments *à la* Gödel to a very weak axiomatic set theory, referring them 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 forms $(\forall x \in y)$ and $(\exists x \in y)$ and are writable in prenex form with at most two alternations of restricted quantifiers (the outermost quantifier being a '$\forall$'). While revisiting their work, we show slightly less weak 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: weak set theories, Gödel incompleteness, essential undecidability

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 [10] or agent-based explainable AI applications (if the agents are to exhibit self-awareness of any form [6]).

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 minimal arithmetic (extremely weak relative to Peano's arithmetic) was proposed by Robinson [21]; 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 an element to a set) was proposed by Vaught [24]. In either case, the proposed axiom system specifies 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: (i) by making all steps needed for the 'arithmetization of syntax' task more transparent; (ii) by basing such a task on formulae of an extremely low syntactic structure. Parlamento and Policriti [18, 19] contributed to these two

Vol. 33, No. 4, © The Author(s) 2023. Published by Oxford University Press.

Advance Access Publication on 31 March 2023 https://doi.org/10.1093/logcom/exad010

amelioration goals, *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* [12]. 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.

Key points of our present contribution are recourse to an alternative definition of ordered pair (see Definition 1); the modelling of certain functions as sets formed by doubletons instead of by ordered pairs (an idea originating from [3] and also exploited in [5]); the introduction of a rather elaborate structure to encode natural numbers (cf. Section 2.1), inspired by the specification of natural numbers in [5], that characterizes numbers and the successor function at the same time in the setting of the full fledged ZF axiomatic system. The structure for natural numbers that we propose overcomes the limitations due to the absence of the infinity axiom by pairing each finite ordinal with the segment of the successor function needed to characterize it. The insights provided by this technique are an essential part of our contribution (cf. Remark 1), and they are a recurrent ingredient in the design of almost all the specifications that lead to the main result of this paper (Theorem 3).

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, under minimal assumptions on the strength of Θ,[1] the procedure returns false 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 forms:

$$\begin{aligned} (\forall\, x_0, \ldots, x_n \in y)\varphi \quad &\overset{\text{Def}}{\longleftrightarrow} \quad \forall x_0 \cdots \forall x_n \big((x_0 \in y \wedge \cdots \wedge x_n \in y) \;\rightarrow\; \varphi \big), \\ (\exists\, x_0, \ldots, x_n \in y)\varphi \quad &\overset{\text{Def}}{\longleftrightarrow} \quad \exists x_0 \cdots \exists x_n \big(x_0 \in y \wedge \cdots \wedge x_n \in y \;\wedge\; \varphi \big). \end{aligned} \tag{1}$$

When a formula involves only restricted quantifiers, it is said to be a Δ_0-formula (see [12]). Furthermore, if it is logically equivalent to some prenex formula with m alternating batches of unbounded quantifiers in the prefix starting with universal quantifiers, then it is said to be of class $(\forall\exists\forall \cdots Q_m)_0$ (where Q_m is $\forall$ when m is odd and $\exists$ otherwise). In [18], it was investigated how, by taking into account 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 virtually any reasonable set theory.[2] The limit found in [18] 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

[1] Such as in any *reasonable* set theory (see [18, Sec. 1]), i.e. a theory that satisfies a subset of the axioms that we propose.
[2] Concerning typical set theories, cf. [23, Sec. 4.6].

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 $v_0, v_1, v_2, \ldots$ of set variables;
- two dyadic relators $\in, =$, designating membership and equality, respectively, as the only predicate symbols of the language;
- a basis of propositional connectives adequate to express all truth functions: for definiteness, we adopt $\neg$ (monadic) and $\rightarrow$ (dyadic), designating respectively negation and material implication;
- the existential quantifier $\exists\, v_i$ and the universal quantifier $\forall\, v_i$, associated with each set variable v_i .

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

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

In the light of axiom **(E)**, stating that distinct sets cannot have the same elements, axiom **(N)** ensures that exactly one empty set exists. Axiom **(W)** and axiom **(L)** induce two natural operations: $(x, y) \overset{\texttt{with}}{\longmapsto} x \cup \{y\}$ and $(x, y) \overset{\texttt{less}}{\longmapsto} x \setminus \{y\}$. 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. [22]); another one is T_s, enhancing T with the *separation* axiom schema (cf. e.g. [11]), namely

$$\textbf{(Sep)} \qquad \forall s \exists r \forall e \Big(\ e \in r \ \ \leftrightarrow \ \ \big(e \in s \wedge \psi \big) \ \Big)^{\forall},$$

where neither of the variables r, s occurs in ψ .[3] Since there are infinitely many formulae ψ, there are infinitely many instances of the schema **(Sep)**: a finite number of instances cannot suffice for an autonomous full-fledged set theory, as was proved in [14]. It should be noted that **(N)** and **(L)** are logically equivalent to instances of **(Sep)**; also the following two sentences, which we will take into account later on, are instances of **(Sep)**:

$$(S_1) \qquad \forall s \exists r \forall e \big(e \in r \leftrightarrow \big(e \in s \wedge (\exists u \in e)\ (\exists v \in u)\ (v \notin e) \big) \big),$$

$$(S_2) \qquad \forall s \exists r \forall e \big(e \in r \leftrightarrow (e \in s \wedge (\exists u \in e)\ (\exists v \in e)\ (u \notin v \wedge v \notin u \wedge v \neq u)) \big)$$

[3] We use $\varphi^{\exists}$, respectively $\varphi^{\forall}$, as shorthands for the existential, resp. universal, closure of any formula φ . To wit, $\varphi^{\forall}$ is the sentence obtained by prefixing φ with a universal quantifier $\forall\, v$ for each variable v occurring free in φ .

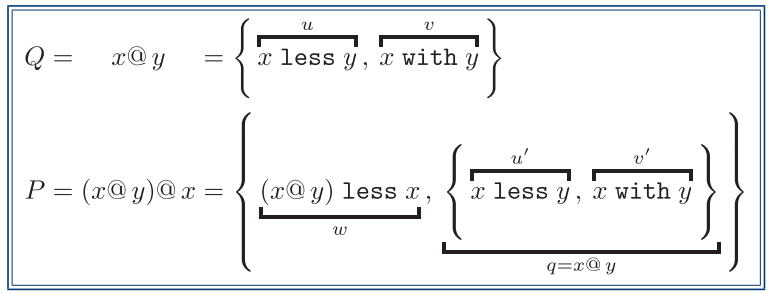

FIGURE 1. Q is a quasi-pair, P an ordered pair; the former notion paves the way to the latter.

A remarkable fact is that each of **(E)**, **(R)**, $(\mathbf{S}_1)$ and $(\mathbf{S}_2)$, as well as the conjunction **(N)**$\wedge$**(W)**$\wedge$**(L)**, can be stated as a three variable sentence: this makes it possible to formalize the axiomatic theory based on those seven sentences in entirely algebraic terms in accordance with [23]; cf. [7, Sec. 6.3.2].

As for T_f , one way of postulating the finitude of all sets in the universe of discourse is the following [23, p.225]:

$$\textbf{(F)} \qquad \forall f\, (\forall t \in f)\, (\exists a \in f)\, (\forall b \in f)\, \big(\quad (\forall d \in b)\, (d \in a) \quad \rightarrow \quad b = a \quad \big)\ .$$

This asserts: every non-empty set f has an $\subseteq$-minimal element a . The set t witnesses that f has elements; $(\forall d \in b)\ (d \in a)$ states the inclusion $b \subseteq a$.

The privileged model for the axiomatic core **(E)** $\wedge$ **(N)** $\wedge$ **(W)** $\wedge$ **(L)** $\wedge$ **(F)** consists of the *pure hereditarily finite sets* ('pure' in the sense that no urelements enter in their construction): these form a universe which is, in essence, an alias of the domain $\mathbb{N}$ of natural numbers and can, accordingly, serve as the support for a theory of computability (cf. [1]). As a matter of fact, a Turing-complete programming language for manipulating the hereditarily finite sets is discussed in [2, Sec. 4.2.1]: some of the primitive constructs of that language (in particular, the constant $\emptyset$, adjunction and removal dyadic operations `with` and `less` and a monadic arbitrary selection operation) implement Skolem functions associated with the said axioms.

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, for instance, the variant $\big\{x, \{x,y\}\big\}$ of Kuratowski's classical ordered pair $\big\{\{x\}, \{x,y\}\big\}$, which one can adopt under **(N)**, **(W)** and **(R)**. This is problematic since, while the extraction of the first component $\varpi_1(p)$ of such a pair $p = \big\{x, \{x,y\}\big\}$ is specifiable by means of an $(\exists)_0$-formula, the extraction of the second projection $\varpi_2(p)$ calls for a formula with at least one quantifier alternation:

$$x = \varpi_1(p) \quad \overset{\text{Def}}{\longleftrightarrow} \quad x \in p \ \wedge\ (\exists q \in p)\, x \in q\,,$$
$$y = \varpi_2(p) \quad \overset{\text{Def}}{\longleftrightarrow} \quad (\exists\, x,\, q \in p)\big(x \in q \ \wedge\ y \in q \ \wedge\ (\forall z \in q)(z = x \ \vee\ z = y)\big).$$

We rely on a definition that works under **(E)**, **(N)**, **(W)** and **(L)**—foundation is not required—introduced in [8]. We use the dyadic operator @ to construct *quasi-pairs*. These will be used in the

formation of *ordered pairs*:

$$x@y \stackrel{\text{Def}}{=} \{x \texttt{ less } y, x \texttt{ with } y\}, \qquad \langle x,y\rangle \stackrel{\text{Def}}{=} (x@y)@x.$$

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

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

We use the first projection extraction on quasi-pairs to obtain the components of ordered pairs, aka 2-tuples, through the following $(\exists)_0$-specifications:[4]

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

For any n, one often specifies n-tuples in terms of 2-tuples. Their projections π^n_i $(0 < i \leq n)$ can then be captured by $(\forall\exists)_0$- and $(\exists)_0$-formulae, respectively; for instance,

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

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

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

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

$$\begin{aligned}
\mathsf{Fnc}(f) \quad &\stackrel{\text{Def}}{\longleftrightarrow} \quad (\forall p \in f)\mathsf{OPair}(p) \wedge \\
&\qquad (\forall p_1, p_2 \in f)(\forall x \in_3 f)(x = \pi^2_1(p_1) = \pi^2_1(p_2) \ \to\ p_1 = p_2).
\end{aligned}$$

We will often write $(\forall x \in \mathsf{dom}\, f)\varphi$ in place of $(\forall p \in f)(\forall x \in_2 p)(x = \pi^2_1(p) \to \varphi)$ and $(\forall y \in \pi^2_1[\mathsf{dom}\, f])\psi$ in place of $(\forall x \in \mathsf{dom}\, f)(\forall y \in_5 f)\big(y = \pi^2_1(x) \ \to\ \psi\big)$ when f is a function and we want to quantify over the first projections of the elements of its domain. In general, to improve readability, these and alike compact forms of restricted quantification will be used, their precise

[4] In specifying projections, it would be pointless to insist that the argument must be an OPair ; consequently, rigorously speaking, we are abbreviating a dyadic relation rather than a monadic function.

specifications sometimes being left as understood. We will also make use of the following function-related notions (cf. [18]), which properly work provided that f is a function:

$$\begin{array}{rcll}
x \in \mathsf{dom}\, f & \overset{\text{Def}}{\longleftrightarrow} & (\exists p \in f)\, x = \pi_1^2(p)\,, & (\exists)_0 \\
d = \mathsf{dom}\, f & \overset{\text{Def}}{\longleftrightarrow} & \big((\forall x \in d)\, x \in \mathsf{dom}\, f\big) \ \wedge\ \big((\forall x \in \mathsf{dom}\, f)\, x \in d\big)\,, & (\forall\exists)_0 \\
y \in \mathsf{ran}\, f & \overset{\text{Def}}{\longleftrightarrow} & (\exists p \in f)\, y = \pi_2^2(p)\,, & (\exists)_0 \\
y = f(x) & \overset{\text{Def}}{\longleftrightarrow} & (\exists p \in f)\, \big(x = \pi_1^2(p) \ \wedge\ y = \pi_2^2(p)\big)\,, & (\exists)_0 \\
v \in f(x) & \overset{\text{Def}}{\longleftrightarrow} & (\exists y \in \mathsf{ran}\, f)\big(y = f(x) \ \wedge\ v \in y\big)\,, & (\exists)_0 \\
\mathsf{Fun}(f(x)) & \overset{\text{Def}}{\longleftrightarrow} & (\forall y \in \mathsf{ran}\, f)\big(y = f(x) \to \mathsf{Fun}(y)\big)\,, & (\forall\exists)_0 \\
v \in \mathsf{dom}\, f(x) & \overset{\text{Def}}{\longleftrightarrow} & (\exists y \in \mathsf{ran}\, f)\big(y = f(x) \ \wedge\ v \in \mathsf{dom}\, y\big)\,, & (\exists)_0 \\
f(x) = g(y) & \overset{\text{Def}}{\longleftrightarrow} & (\forall v \in \mathsf{ran}\, f)(\forall w \in \mathsf{ran}\, f)\big(\big(v = f(x) \ \wedge\ w = g(y)\big) \ \to\ v = w\big), & (\forall)_0 \\
\overline{0} = \mathsf{dom}\, f & \overset{\text{Def}}{\longleftrightarrow} & (\forall x \in \mathsf{dom}\, f)(x \in \mathsf{dom}\, f \to x \neq x), & (\forall)_0 \\
\overline{0} \in \mathsf{dom}\, f & \overset{\text{Def}}{\longleftrightarrow} & (\exists x \in \mathsf{dom}\, f)(x \in \mathsf{dom}\, f \ \wedge\ y = \overline{0}), & (\exists)_0 \\
v \in f(\overline{0}) & \overset{\text{Def}}{\longleftrightarrow} & (\forall y \in \mathsf{ran}\, f)(\forall x \in \mathsf{dom}\, f)(x = \overline{0} \ \wedge\ y = f(x) \ \to v \in y). & (\forall)_0
\end{array}$$

Now and then other akin predicates, not listed here, will show up.[5]

After von Neumann [15], it is convenient to think of natural numbers as simply being the finite ordinals. While the definition of ordinals—see below—is $(\forall)_0$ and hence has a very low syntactic complexity (in the sense explained in the Introduction), expressing their finitude in weak theories requires more effort. We put

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

so that s^+ and s^- denote, resp., the successor $s \cup \{s\}$ and the predecessor of s.

Under **(E)**, **(N)**, **(W)**, **(L)** and **(R)**, natural numbers are expressed by the following $(\forall\exists\forall)_0$-specifications (cf. [20]):

$$\begin{array}{rcl}
\mathsf{Trans}(X) & \overset{\text{Def}}{\longleftrightarrow} & (\forall v \in X)\ (\forall u \in v)\ u \in X\,, \\
\mathsf{Ord}(X) & \overset{\text{Def}}{\longleftrightarrow} & \mathsf{Trans}(X) \ \wedge\ (\forall u, v \in X)\ \big(u \in v \ \vee\ v \in u \ \vee\ v = u\big)\,, \\
\mathsf{Num}_1(X) & \overset{\text{Def}}{\longleftrightarrow} & (\forall t \in X)\ (\exists y \in X)\ (\forall u \in X)\ (u = y \vee u \in y) \quad \wedge \\
 & & (\forall y \in X)\ (\forall t \in y)\ (\exists z \in y)\ (\forall u \in y)\ (u = z \vee u \in z) \quad \wedge \quad \mathsf{Ord}(X)\,.
\end{array}$$

In T_s, more specifically thanks to the above-cited instances $(\mathbf{S}_1)$ and $(\mathbf{S}_2)$ of the separation axiom schema **(Sep)**, the latter can be simplified into the following simpler $(\forall\exists\forall)_0$-formula:

$$\begin{array}{rcll}
\mathsf{Num}_2(X) & \overset{\text{Def}}{\longleftrightarrow} & (\forall y \in X^+)\ \big(y = \varnothing \vee (\exists z \in X)\ z^+ = y\big) \wedge & \\
 & & (\forall u, v \in X)\ (u \in v \vee v \in u \vee v = u)\,. & (\dagger)
\end{array}$$

The second quantifier alternation in the definiens of this (†) is caused by the occurrence $z^+ = y$ within it. In the next section, we will get rid of this alternation by resorting to a less demanding yet functionally equivalent specification of the successor function.

[5] One should view the notations $x \in \mathsf{dom}\, f$ and $y \in \mathsf{ran}\, f$ as designating dyadic relations over the universe of sets, devoid of any 'ontological commitment' about $\mathsf{dom}\, f$ and $\mathsf{ran}\, f$. In fact, under weak axioms neither $\mathsf{dom}\, f$ nor $\mathsf{ran}\, f$ are guaranteed to designate entities that one can view as sets inside the theory. On the other hand, $d = \mathsf{dom}\, f$ ensures the status of a genuine set to the class $\mathsf{dom}\, f$.

Note that in T_f the property 'being a natural number' simply amounts to Ord; hence, its definiens 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 strengthen the structure of this property in the manner discussed next.

2.1 Natural numbers

In [4], a $(\forall\exists)_0$-characterization of the first limit ordinal is framed in the full-fledged ZF theory, the driving idea being that the set of natural numbers should be characterized together with the (infinite) successor function on natural numbers. However, in the present setting, the same approach is not viable, as no guarantee of existence of an infinite set comes from our axiomatic core. Nevertheless, it is useful to note that it is sufficient to require a bit less to enable some important reductions in $(\forall\exists)_0$-complexity. Indeed, a streamlined definition of the natural numbers predicate that models each number as an object pairing a finite ordinal with the restriction of the successor function to it, allows us to characterize the successor of each number by means of an $(\exists)_0$-formula, instead of a $(\forall)_0$ one. This approach adds flexibility w.r.t. (†), thanks to the general idea of 'storing information' inside functions so that it can be 'accessed' by means of an $(\exists)_0$-formula (cf. Remark 1 below); here is one major ingredient contributed by this paper.

REMARK 1
Suppose that $\mathsf{D}(x)$ is an $(\exists)_0$ predicate and $\mathsf{F}(x,y)$ is some $(\forall\exists)_0$ functional predicate over the class where $\mathsf{D}(x)$ holds[6] containing at least one restricted universal quantifier. Of course, writing F under the scope of a restricted existential quantifier yields a formula which is not $(\forall\exists)_0$. However, for any variable f (either universally or existentially quantified),

$$\mathsf{Fnc}(f) \wedge (\forall x \in \mathsf{dom}f)(\forall y \in \mathsf{ran}f)\big[y = f(x) \rightarrow \mathsf{F}(x,y)\big]$$

is a $(\forall\exists)_0$-formula that characterizes a function f which is a restriction of F. Let us call this formula φ_f. For any set X such that $D(x)$ holds for every $x \in X$ (so F is defined on X), if it is possible to express that every element of X is in the domain of f by means of a $(\forall\exists)_0$-formula ψ_f, then $\mathsf{F}(x,y)$ is expressible by an $(\exists)_0$ formula for the elements in X; thus, making it possible to write $(\forall\exists)_0$-formulae where the predicate occurs under the scope of a restricted existential quantifier.

Our next example illustrates how to apply the technique described above.

EXAMPLE 1
Consider again the predicate

$$y = x^+ \quad \overset{\text{Def}}{\longleftrightarrow} \quad (\forall z \in y)(z \in x \vee z = x) \wedge x \in y \wedge (\forall z \in x)z \in y,$$

as defined on finite ordinals. Suppose that n is some finite ordinal. While expressing $\mathsf{F}(x,y) \overset{\text{Def}}{\longleftrightarrow} y = x^+$ for ordinals that satisfy $\mathsf{D}(x) \overset{\text{Def}}{\longleftrightarrow} x \in n$ would normally be possible only by means of a $(\forall)_0$-formula, if we characterize some variable f in the following way:

$$\mathsf{Fnc}(f) \wedge (\forall x \in \mathsf{dom}\, f)(\forall y \in \mathsf{ran}f)\big[y = f(x) \rightarrow y = x^+\big]$$

[6] *Functional predicate* over D means that $\forall x \exists y \forall y' [\mathsf{D}(x) \wedge F(x,y') \Leftrightarrow y' = y]$.

and

$$(\forall m \in n)\ m \in \mathsf{dom}\, f,$$

then $y = x^+$ for ordinals $x \in n$ turns out to be expressible by the $(\exists)_0$-formula

$$y = f(x).$$

Much as in [4], we think of the successor function as a set of doubletons (proper), as, since the function is monotonic, it is always possible to discern which one of the elements of each unordered pair is part of the domain. We think of natural numbers as specially constrained unordered pairs, where one of the elements is the finite ordinal representation of the number, and the other one is the restriction of the successor function on it. The following predicates:

$$\begin{aligned}
\mathsf{NPair}(p) &\overset{\text{Def}}{\longleftrightarrow} [((\exists x \in p)\top \wedge (\forall x \in p)(\forall y \in x)\bot) \vee (\exists u, f \in p)(\exists q \in f)(u \in q)] \wedge \\
&\quad (\forall v_1, v_2, v_3 \in p)(v_1 = v_2 \vee v_2 = v_3 \vee v_1 = v_3) \\
x \mathrel{\overline{\in\in}} p &\overset{\text{Def}}{\longleftrightarrow} x \in p \wedge [(\exists f \in p)(\exists q \in f)(u \in f) \vee ((\forall z \in x)\bot \wedge (\forall z \in p) z = x)] \\
x \mathrel{\underline{\in\in}} p &\overset{\text{Def}}{\longleftrightarrow} y \in p \wedge [(\exists x \in p)(x \mathrel{\overline{\in\in}} p \wedge y \neq x) \vee ((\forall z \in y)\bot \wedge (\forall z \in p) z = y)],
\end{aligned}$$

serve as a base for our definition of natural numbers.[7] The first one characterizes unordered pairs where one of the two elements belongs to one of the elements of the other, and the last two characterize the first and second projection, respectively (which will concretely be the natural number and the restriction of the successor function, respectively). We are now ready to present yet another characterization of the natural number predicate—henceforth, our official one:

$$\begin{aligned}
\mathsf{Num}(P) \overset{\text{Def}}{\longleftrightarrow}\ & \mathsf{NPair}(P) \wedge \\
& (\forall f \mathrel{\underline{\in\in}} P)(\forall q \in f)\big[(\forall v_1, v_2, v_3 \in q)(v_1 = v_2 \vee v_2 = v_3 \vee v_1 = v_3) \wedge \\
& \qquad (\exists v_1, v_2 \in q)(v_1 \in v_2)\big] \wedge \\
& (\forall f \mathrel{\underline{\in\in}} P)(\forall q \in f)(\forall x, y \in q)(x \in y \rightarrow (\forall z \in y)(z \in x \vee z = x)) \wedge \\
& (\forall n \mathrel{\overline{\in\in}} P)(\forall u \in n)(\exists f \mathrel{\underline{\in\in}} P)(\exists q \in f)(\exists y \in q)(u \in q \wedge u \in y) \wedge \\
& (\forall n \mathrel{\overline{\in\in}} P)(\forall y \in n^{++})(\forall f \mathrel{\underline{\in\in}} P)\ \big(y = \varnothing \vee (\exists z \in n)\ z^{++} = y\big) \wedge \\
& (\forall n \mathrel{\overline{\in\in}} P)(\forall u, v \in n)\ (u \in v \vee v \in u \vee v = u),
\end{aligned}$$

where $(\forall y \in n^{++})\varphi(y)$ and $z^{++} = y$ abbreviate and

$$\varphi(n) \wedge (\forall y \in n)\varphi(y) \quad \text{and} \quad (\exists q \in f)(z \in q \wedge y \in q \wedge z \in y),$$

respectively. The conditions in the specification assert that

1. a natural number is an unordered pair P such that $\mathsf{NPair}(P)$;
2. each element of the second projection f of the said P is a doubleton one of whose elements is a member of the other, i.e. f is an $\in$-monotonic unordered map;
3. for each ordinal u smaller than the first projection n, an unordered pair belonging to f consists of u and its successor;
4. both conditions that define finite ordinals according to (†) hold for n.

An $(\exists)_0$-characterization of item 4 was possible thanks to the fact that $z^+ = y$ can be expressed by an $(\exists)_0$-formula in this context (cf. Example 1, Remark 1 and the definition of $z^{++} = y$). This insight that, under the circumstances in Remark 1, sets characterized by $(\forall\exists)_0$-formulae can be 'stored' in

[7] The symbols $\top$ and $\bot$ can be characterized by $(\forall x)x = x$ and its negation, respectively.

maps and 'accessed' where needed with just an $(\exists)_0$-formula is one of the main Δ_0-complexity reduction techniques used throughout the paper.

All of the following definitions can be straightforwardly re-adapted to use this last $(\forall\exists)_0$-formula; however, it is possible to use the ideas developed in the specification above to characterize finite ordinals (instead of pairs containing a finite ordinal and a restriction of the successor function) in all the formulae in this paper, while writing them as Σ_1 formulae with $(\forall\exists)_0$ matrix. Indeed, the reason why the pair characterization seems necessary is that the presence of the successor function allows to reduce the complexity of the finite ordinal specification. Hence, if there was some constant **s** representing the successor function on finite ordinals, the specification of natural number would become

$$\begin{array}{rcl}\mathsf{Num}_{\mathbf{s}}(n) & \overset{\text{Def}}{\longleftrightarrow} & (\forall y \in n^{++})\left(y = \varnothing \vee (\exists z \in n)\, z^{++} = y\right) \wedge \\ & & (\forall u, v \in n)\,(u \in v \vee v \in u \vee v = u),\end{array}$$

where $z^{++} = y$ stands for

$$(\exists q \in \mathbf{s})(z \in q \wedge y \in q \wedge z \in y),$$

i.e. a $(\forall\exists)_0$-formula. This comes at no cost, as in all formulae that we consider, it is possible to characterize the restriction of the successor function up to the highest finite ordinal defined without altering the Δ_0-complexity of the formula. Indeed, for each Σ_1-formula with $(\forall\exists)_0$-matrix

$$\exists x_1 \cdots \exists x_n \varphi\,,$$

consider the Σ_1-formula

$$\begin{array}{rll}\exists\mathbf{s}\exists x_1 \cdots \exists x_n & \big((\forall p \in \mathbf{s})(\forall v_1, v_2, v_3 \in p)(v_1 = v_2 \vee v_2 = v_3 \vee v_1 = v_3) & \wedge \\ & (\forall p \in \mathbf{s})(\forall x, y \in p)(x \neq y \Rightarrow x \in y \vee y \in x) & \wedge \\ & (\forall p \in \mathbf{s})(\forall x, y \in p)(x \in y \Rightarrow (\forall z \in y)(z \in x \vee z = x)) & \wedge \\ & (\forall p \in \mathbf{s})(\forall x, y \in p)(x \in y \Rightarrow (\forall z \in x)(\exists q \in \mathbf{s})(\exists w \in q) & \\ & \qquad (z \in q \wedge z \in w)) & \wedge \\ & \psi \wedge \varphi\big), & \end{array}$$

where ψ asserts that all finite ordinals occurring in φ are in the domain of **s**. Whenever ψ is $(\forall\exists)_0$, the whole formula is clearly Σ_1 with $(\forall\exists)_0$-matrix. In all the formulae needed to prove the main result of this paper, ψ is $(\forall\exists)_0$.

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:

$$\begin{array}{rcl} x = \overline{0} & \overset{\text{Def}}{\longleftrightarrow} & \exists x_0(x_0 = \varnothing \wedge x = x_0) \\ x = \overline{n+1} & \overset{\text{Def}}{\longleftrightarrow} & (\exists x_0) \cdots (\exists x_n)\,(x_0 = \varnothing \ \wedge\ \textstyle\bigwedge_{i=1}^{n} x_i = x_{i-1}^{+} \ \wedge\ x = x_n^{+}). \end{array}$$

REMARK 2
The definition of $x = \overline{0}$ has not been written as just $x = \varnothing$, a $(\forall\exists)_0$-formula. The reason is that, such as with **s**, it is possible to introduce an existentially quantified unrestricted variable x_0 which is forced to represent the empty set, so that no quantifiers are needed at all to express $x = \overline{0}$. Literals of this form should henceforth be read in this way that does not introduces universal quantifiers.

The following claims are plain (cf. [18]).

LEMMA 1
In T, the following are provable:

1. $(\overline{n})^+ = \overline{n+1}, x = \overline{n} \wedge y = x^+ \rightarrow y = \overline{n+1}$;
2. $x^+ = \overline{n+1} \rightarrow x = \overline{n}, \mathsf{Num}(x) \rightarrow x \neq x$;
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. $\mathsf{Num}(\overline{0}), \mathsf{Num}(\overline{n}), \mathsf{Num}(x) \rightarrow \mathsf{Num}(x^+), \forall x(\mathsf{Num}(x) \rightarrow x \neq \overline{0})$;
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 ([18]).
Every total recursive n-ary function f on $\mathbb{N} \stackrel{\text{Def}}{=} 0, 1, 2, \ldots$ is *strongly representable* in T', in the sense that there is a formula φ such that for $k_1, \ldots, k_n, k \in \mathbb{N}$:

- $f(k_1, \ldots, k_n) = k$ implies $T' \vdash \varphi(\overline{k}_1, \ldots, \overline{k}_n, \overline{k})$ and
- $T' \vdash \exists x\, \forall y \left(\varphi(\overline{k}_1, \ldots, \overline{k}_n, y) \leftrightarrow y = x \right)$.

3 Codes

We will revamp the original definition of code in [18] 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 will allow us to restrict all universal quantifiers in Definitions 2 and 3, while the remaining ones are triples that encode the nodes of the tree. Each triple has the node type as its first component, and either two leaves (variable nodes) or pointers to nodes previously appearing in the sequence as second and third components. The last triple in the sequence is considered to be the root of the tree. As will be clear, for each formula φ, there is a countable infinity of code sequences encoding φ.

In order to get a simple definition, we will make use of *un*ordered functions in our definitions.

DEFINITION 2 (Code sequence).

$$\begin{aligned}\mathsf{Seq}_C(f) \quad \stackrel{\text{Def}}{\longleftrightarrow} \quad & \mathsf{Num}(f \restriction \overline{0}) \wedge \big(\forall x \in_2 f\big)\big(x \in \mathsf{dom}_C f \rightarrow (\exists\, w \in f \restriction \overline{0})\, x \in w\big) \wedge \\ & (\forall x \in f \restriction \overline{0})\big(x \in \mathsf{dom}_C f \rightarrow (\forall y \in x)\, y \in \mathsf{dom}_C f\big) \wedge \\ & (\forall p \in f)\big((\forall x, y, z \in p)(x = y \vee y = z \vee x = z) \wedge (\exists x, y \in p)(x \neq y)\big) \wedge \\ & (\forall p \in f)(\forall x, y \in p)\big[x \neq y \wedge x \neq \overline{0} \wedge y \neq \overline{0} \rightarrow \\ & \qquad\qquad y \notin x \wedge x \notin y \wedge \big(x \in f \restriction \overline{0} \vee y \in f \restriction \overline{0}\big)\big],\end{aligned}$$

where

$$\begin{aligned} x \in \mathsf{dom}_C f \quad &\overset{\text{Def}}{\longleftrightarrow} \quad (\exists p \in f)(\exists y \in p)[\, x \in p \ \wedge\ x \neq y \ \wedge\ (x \in y \ \vee\ \overline{0} \in x)\,], \\ y = f \restriction x \quad &\overset{\text{Def}}{\longleftrightarrow} \quad x \in \mathsf{dom}_C f \ \wedge\ (\exists p \in f)\, x, y \in p \ \wedge\ y \neq x \ \wedge\ (x \in y \to x = \overline{0}), \\ (\forall x \in f \restriction \overline{0})\varphi \quad &\overset{\text{Def}}{\longleftrightarrow} \quad (\forall p \in f)(\forall z \in p)(\forall y \in p)(z = \overline{0} \wedge y = f \restriction z \to (\forall x \in y)\varphi), \end{aligned}$$

so that $x \in \mathsf{dom}_C f$ means 'x is in the unordered domain of the code sequence f' and $f \restriction x$ is the image of x under f.

The fourth condition in the definiens of $\mathsf{Seq}_C(f)$ forces f to be made up of doubletons proper, while the fifth one establishes that each such doubleton in f, save $\overline{0}$ paired with its image, is made of a number and a non-number. The first three conjuncts state that the first element (*height*, $f \restriction \overline{0}$, of the sequence) is a natural number and that it exceeds the domain (*length*) of the sequence.

These definitions have the desired meanings; in fact, as the length of any code sequence is nonnull, its height must exceed 1. Hence, $\overline{0} \in f \restriction \overline{0}$ whenever $\mathsf{Seq}_C(f)$ holds. The basic principle that makes the above constructs have the desired meaning is that, thanks to regularity, it is always possible to prove that two natural numbers are comparable by membership. 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:

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

DEFINITION 3 (Code).

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

We already noted that Triple is $(\forall\exists)_0$ and the last two conjuncts in the consequent 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, while 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

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

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

$$\big[\pi_3(f_m) <_S \pi_3(g_m)\big] \vee \big[\pi_3(f_m) = \pi_3(g_m) \wedge \pi_2(f_m) < \pi_2(g_m)\big] \vee \big[\pi_3(f_m) = \pi_3(g_m) \wedge \pi_2(f_m) = \pi_2(g_m) \wedge \pi_1(f_m) < \pi_1(g_m)\big].$$

Here, by $<_S$, we are denoting a strict total order on the collection of symbols. Since the projections of f can be extracted by means of $(\exists)_0$-formulae, and $\mathsf{dom}_C f = \mathsf{dom}_C g$ can be stated as a $(\forall\exists)_0$-formula, the overall syntactic complexity mostly relies on $<_S$. Under suitable definitions of Symbol and $<_S$ (e.g. if we identify symbols with natural numbers up to some finite bound n), $f \leq_C g$ turns out to be a $(\forall\exists)_0$-formula. The following proposition is trivially proved by contradiction.

LEMMA 2
In T, since $<_S$ is a total order, also $\leq_C$ is total.

We can also explicitly define the strict variant of $\leq_C$ by means of 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 next specify a successor function that will enable an enumeration of all 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 shall be written as

$$g = \mathsf{Next}_C(f) \overset{\text{Def}}{\longleftrightarrow} (C_1) \vee (C_2) \vee (C_3).$$

3.0.1 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 base-c number. The variable v is conveniently used in the antecedent of the implications in order to be able to write a $(\exists)_0$-formula.

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

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:

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

We are now ready to produce a compact $(\forall\exists)_0$-specification of Condition (C_1):

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

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

3.0.2 Condition (C_2):

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

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

3.0.3 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 with the code $\{\{0,3\},\{1,\langle 0,0,0\rangle\}\}$ and through successive applications of Next_C, we can enumerate all codes; things are so as, given a length and a height, only a finite amount of codes are endowed with those length and height. The bottom code, $\overline{0}_C$, is characterized by the $(\forall\exists)_0$-formula

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

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 [18], we will not provide details.

LEMMA 4
In T:

$$(a)\quad \forall x \forall y \big(\mathsf{Cod}(x) \wedge \mathsf{Next}_C(x) = y \rightarrow \mathsf{Cod}(y)\big);$$

$$(b)\quad \forall x \forall y \forall z \big(\mathsf{Cod}(x) \wedge \mathsf{Cod}(y) \wedge \mathsf{Cod}(z) \wedge x \leq_C z \wedge z \leq_C y \wedge y = \mathsf{Next}_C(x) \rightarrow z = x \vee z = y\big);$$

$$(c)\quad \forall x \forall y \big(\mathsf{Cod}(x) \wedge \mathsf{Cod}(y) \wedge \mathsf{Next}_C(x) = y \rightarrow x <_C y\big);$$

$$(d)\quad \forall x \forall y \forall z \big(\mathsf{Cod}(y) \wedge \mathsf{Next}_C(y) = x \wedge \mathsf{Next}_C(y) = z \rightarrow x = z\big).$$

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

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

LEMMA 5
For each natural number k, in T:

$$(a)\quad \overline{(k+1)}_C = \mathsf{Next}_C(\overline{(k)}_C);$$

$$(b)\quad x = \overline{(k)}_C \wedge y = \mathsf{Next}_C(x) \rightarrow = \overline{(k)}_C;$$

$$(c)\quad \forall x(\mathsf{Cod}(x) \wedge \overline{(k+1)}_C = \mathsf{Next}_C(x) \rightarrow x = \overline{(k)}_C$$

$$(d)\quad \mathsf{Cod}(\overline{(k)}_C);$$

$$(e)\quad \forall x\big(\mathsf{Cod}(x) \rightarrow \neg <_C \overline{0}_C\big)$$

$$(f)\quad \forall x\big(\mathsf{Cod}(x) \rightarrow (x \leq_C \overline{(k)}_C \leftrightarrow x <_C \overline{(k+1)}_C)\big)$$

$$(g)\quad \forall x\big(\mathsf{Cod}(x) \rightarrow (x <_C \overline{(k+1)}_C \leftrightarrow x = \overline{0}_C \vee \ldots \vee x = \overline{(k)}_C)\big);$$

$$(h)\quad \forall x\big(\mathsf{Cod}(x) \rightarrow (x <_C \overline{(k)}_C \vee x = \overline{(k)}_C \vee \overline{(k)}_C <_C x)\big)$$

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 are $(\exists)_0$-formulae. 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).

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

This clearly is a $(\forall\exists)_0$-formula. 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 $\varphi \rightarrow \bot$; accordingly, $\exists v\varphi$ will stand for $\big(\forall v(\varphi \rightarrow \bot)\big) \rightarrow \bot$.

We adopt as logical axioms for first-order predicate calculus with equality the following schemata:[8]

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

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 (A5) are more problematic. Formulae that are instances of the schema (A3) are captured by the following predicate $\mathsf{A3}$:

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

One can proceed similarly with (A5). 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

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

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.

[8] Axiom A1 alone encodes propositional logic *à la* Meredith [13, p. 39].

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 two '*CleanCopy*' predicates holding when f is a left, resp., a right copy of g:

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

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

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

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:

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

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 consequent 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 $(f \restriction \bar{0})^{-}$); and (iii) a list of indices between 1 and $f \restriction \bar{0}$ 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.

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

where the formula Ψ will be defined below. Clearly, 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.

1. $\mathsf{Axiom}(f_i)$. Clearly, to recognize axiom (A5), 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.
2. Modus Ponens:

$$\begin{aligned}
(\exists j_1, j_2 \in i)(f(\bar{0}) \wedge f(\bar{0}) \leq j_2 \wedge \mathsf{Symbol}_{\Rightarrow}(\pi_1(\mathsf{last}(\pi_1 f(f_{j_1}))) \wedge \\
\pi_1(f(f_i)) = \pi_3(f(f_{j_1})) \wedge \pi_1(f(f_{j_2})) = \pi_2(f(f_{j_1}))).
\end{aligned}$$

3. Universal generalization:

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

4. Rename resolution:

$$\begin{aligned}
&(\forall k \in \mathsf{dom}\, \pi_3(f(f_{p_i})))(\forall t \in \mathsf{ran}\pi_3(f(f_{p_i})))(\forall j \in f(\bar{0})) \\
&\left[j = \mathsf{rfrom}(\mathsf{last} f(f_i)) \wedge t = \pi_3(f(f_{p_i}))(k) \to \psi\right] \wedge f(\bar{0}) \leq p_i,
\end{aligned}$$

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 is a (big) formula (cf. Section 4.1 below) 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:

$$\mathsf{Proof}(p,x) \quad \overset{\text{Def}}{\longleftrightarrow} \quad \mathsf{Quintuple}(p) \ \wedge \ \mathsf{Proof}'\big(\pi_1(p), x, \pi_2(p), \pi_3(p), \pi_4(p), \pi_5(p)\big).$$

Notice that it is easy to express that all the natural numbers occurring in a proof belong to the successor function **s** introduced in Section 2.1. Indeed, each of such numbers is characterized by a $(\forall)_0$-formula; therefore, it is possible to state that it belongs to the domain of **s** by using a $(\forall)_0$-formula.

4.1 Expanded version of the rename resolution disjunct

Here, follows the expanded version of the formula ψ in the 'Rename Resolution' disjunct in Proof':

$$\begin{aligned}&(\forall k \in \mathsf{dom}\ \pi_3(f(f_{p_i})))(\forall t \in \mathsf{ran}\pi_3(f(f_{p_i})))(\forall j \in f(\overline{0}))\\ &(j = \mathsf{rfrom}(\mathsf{last}f(f_i)) \ \wedge \ t = \pi_3(f(f_{p_i}))(k) \to \psi') \wedge f(\overline{0}) \leq p_i,\end{aligned}$$

where ψ' is the conjunction of the following formulae representing the different possible cases:

- $\mathsf{Symbol}_{R_j}(\pi_1(\mathsf{last}(\pi_1(f(f(p_i))))))$,
- $\mathsf{Symbol}_{\Rightarrow}(\pi_1(t)) \to (\pi_1(f(f_i)))(k) = t$,
- $\mathsf{Symbol}_{\forall}(\pi_1(t)) \ \wedge \ \pi_2(t) = j \ \to \ (\pi_1(f(f_i)))(k) = \langle \pi_1(t), j, \pi_3(t)\rangle$,
- $\mathsf{Symbol}_{\forall}(\pi_1(t)) \ \wedge \ \pi_2(t) \neq j \ \to \ (\pi_1(f(f_i)))(k) = t$,
- $\mathsf{Symbol}_{=}(\pi_1(t)) \ \wedge \ \pi_2(t) = j \ \wedge \ \pi_3(t) = j \ \to (\pi_1(f(f_i)))(k) = \langle \pi_1(t), \pi_2(\mathsf{last}\pi_1(f(f_i))),$ $\pi_2(\mathsf{last}\pi_1(f(f_i)))\rangle$,
- $\mathsf{Symbol}_{=}(\pi_1(t)) \wedge \pi_2(t) = j \wedge \pi_3(t) \neq j \ \to (\pi_1(f(f_i)))(k) = \langle \pi_1(t), \pi_2(\mathsf{last}\pi_1(f(f_i))), \pi_3(t)\rangle$,
- $\mathsf{Symbol}_{=}(\pi_1(t)) \wedge \pi_2(t) \neq j \wedge \pi_3(t) = j \ \to (\pi_1(f(f_i)))(k) = \langle \pi_1(t), \pi_2(t), \pi_2(\mathsf{last}\pi_1(f(f_i)))\rangle$,
- $\mathsf{Symbol}_{=}(\pi_1(t)) \ \wedge \ \pi_2(t) \neq j \ \wedge \ \pi_3(t) \neq j \ \to (\pi_1(f(f_i)))(k) = t$,
- $\mathsf{Symbol}_{\in}(\pi_1(t)) \ \wedge \ \pi_2(t) = j \ \wedge \ \pi_3(t) = j \ \to (\pi_1(f(f_i)))(k) = \langle \pi_1(t), \pi_2(\mathsf{last}\pi_1(f(f_i))),$ $\pi_2(\mathsf{last}\pi_1(f(f_i)))\rangle$,
- $\mathsf{Symbol}_{\in}(\pi_1(t)) \wedge \pi_2(t) = j \wedge \pi_3(t) \neq j \ \to (\pi_1(f(f_i)))(k) = \langle \pi_1(t), \pi_2(\mathsf{last}\pi_1(f(f_i))), \pi_3(t)\rangle$,
- $\mathsf{Symbol}_{\in}(\pi_1(t)) \wedge \pi_2(t) \neq j \wedge \pi_3(t) = j \ \to (\pi_1(f(f_i)))(k) = \langle \pi_1(t), \pi_2(t), \pi_2(\mathsf{last}\pi_1(f(f_i)))\rangle$,
- $\mathsf{Symbol}_{\in}(\pi_1(t)) \ \wedge \ \pi_2(t) \neq j \ \wedge \ \pi_3(t) \neq j \ \to (\pi_1(f(f_i)))(k) = t$.

5 Essential Undecidability

What follows presupposes an essential preliminary step for an *arithmetization of the syntax*, namely that a map $\varphi \mapsto \ulcorner\varphi\urcorner$ has been implemented (cf. [16]) sending every formula into a natural number; a number whence one can retrieve a formula φ' such that $\varphi' \dashv\vdash \varphi$. 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 $\varphi_{\mathtt{ind}}(c,k)$ (f is meant to be a function $\mathsf{Cod} \to \mathbb{N}$):

$$\begin{aligned}\exists f\Big(\quad &\mathsf{Fnc}(f) \ \wedge \ \mathsf{Cod}(c) \ \wedge \ \mathsf{Num}(k) \ \wedge \ \mathsf{dom}\ f = k^+ \ \wedge f(\overline{0}) = \overline{(0)}_C \wedge\\ &(\forall z \in k)\big(\mathsf{Cod}(f_{z^+}) \ \wedge f_{z^+} = \mathsf{Next}_C(f_z)\big) \ \wedge f_k = c \ \Big).\end{aligned}$$

PROOF. Clearly, if k is the index of a code c, then $\varphi_{\mathtt{ind}}(c,k)$ holds. We must prove that if c is the k-th code following the $<_C$ order, then the formula holds on c only when the second argument is k

(so that $\varphi_{\mathtt{ind}}$ represents a function), namely

$$\forall y\big(\varphi_{\mathtt{ind}}(c,y) \to y = k\big).$$

We proceed by induction on k. When $k = 0$, then $c = \overline{(0)}_C$. Suppose there is y such that $\varphi_{\mathtt{ind}}(\overline{(0)}_C, y)$ for some function f, that $y \neq 0$, and

$$\forall y\big(\varphi_{\mathtt{ind}}(\overline{(k)}_C, y) \to y = k\big).$$

Therefore, y has some predecessor z such that $\overline{(0)}_C = \mathsf{Next}_C(f_z)$ holds, in view of $\varphi_{\mathtt{ind}}(\overline{(0)}_C, y)$. But this is untenable, as it would mean that $f_z <_C \overline{(0)}_C$, and we already proved that this is impossible.

Let us now consider the inductive case. Suppose that $\varphi_{\mathtt{ind}}(\overline{(k+1)}_C, y)$ holds and $y \neq k$. As clearly $\overline{(k+1)}_C \neq \overline{(0)}_C$, then $y \neq 0$, and, hence, y has a predecessor z. Consider now

$$h = f \ \mathtt{less}\ \langle y, \overline{(k+1)}_C\rangle.$$

Such h is guaranteed to exist thanks to axiom **(L)**. By inductive hypothesis, we have that $z = k$; thus, $y = k + 1$. □

LEMMA 7

The predicate $\mathsf{SubstV}(c, v, d)$ that holds when the formula with code d is obtained from the formula with code c substituting the first variable (lowest index) with the variable of index v is strongly represented in T' by a Σ_1-formula with a $(\forall\exists)_0$-matrix $\varphi_{\mathtt{subv}}(c, v, d)$:

$$\begin{aligned}
&\exists c'\exists m\exists f\\
\Big(\ &(\forall p \in c)(c \in c') \ \wedge\ \mathsf{dom}_C\, c' = \mathsf{dom}_C\, c^{+} \ \wedge\ (\exists i \in \mathsf{dom}_C\, c)\big(\mathsf{VarIn}(m, c_i)\big) \wedge\\
&(\forall i \in \mathsf{dom}_C\, C)(\forall v \in \pi_2(\mathsf{ran}_C\, c))(\mathsf{VarIn}(v, c_i) \to m \leq v) \wedge\\
&(\forall i \in \mathsf{dom}_C\, C)(\forall v \in \pi_3(\mathsf{ran}_C\, c))(\mathsf{VarIn}(v, c_i) \to m \leq v) \wedge\\
&\mathsf{Symbol}_{R_m}(\pi_1(c'(\mathsf{dom}_C\, c))) \ \wedge\ \pi_2(c'(\mathsf{dom}_C\, c)) = v \wedge\\
&\pi_3(c'(\mathsf{dom}_C\, c)) = \mathsf{dom}_C\, c^{-} \ \wedge\ \varphi_{\mathtt{ren}}(c', d, f)\Big),
\end{aligned}$$

where $\varphi_{\mathtt{ren}}(c', d, f)$ is the formula that strongly represents the rename resolution operation (cf. Lemma 10 below) and $\mathsf{VarIn}(i, t)$ holds whenever i is a variable in the triple t (plainly an $(\exists)_0$-formula).

PROOF. It is trivial to note that c' is forced to be the formula represented by c, plus a rename of the variable with lowest index m to the variable v. As we already proved that $\varphi_{\mathtt{ren}}$ resolves the renaming, d has to be the formula code obtained by renaming m to v. □

We will use $\varphi_{\mathtt{subv}}(c, v, d, w)$ to indicate the same formula, where w is a triple containing c', m and f; hence, it will be possible to existentially quantify w in the prefix of the formulae in which we will use $\varphi_{\mathtt{subv}}$. Using a similar technique, we also define a formula that strongly represents *term substitution*. In order to do so, we need to define a formula that strongly represents code concatenation.

LEMMA 8
The function that, given two codes c_1 and c_2, outputs their concatenation d with corrected pointers is strongly representable through a Σ_1-formula with a $(\forall\exists)_0$-matrix $\varphi_{\mathtt{cat}}(c_1, c_2, d)$:

$$
\begin{aligned}
&\exists f \exists g \exists n_1 \exists n_2 \exists n \\
\Big(\;& \mathsf{Fnc}(f) \wedge \mathsf{Fnc}(g) \wedge \\
& n_1 = \mathsf{dom}_C\, c_1 \wedge n_2 = \mathsf{dom}_C\, c_2 \wedge \mathsf{dom}\, g = n_2^+ \wedge g(\bar{0}) = n_1 \wedge \\
& (\forall i \in n_2)(g_{i^+} = g_i^+) \wedge n = g_{n_2} \wedge \mathsf{dom}\, f = n_1^+ \wedge f_{n_1} = d \wedge \\
& (\forall i \in n_2)(\forall j \in n_1)(\mathsf{Cod}(f_i) \wedge \mathsf{dom}_C\, f_i = n \wedge (f_i)_j = (c_1)_i) \wedge \\
& (\forall j \in n)\big(j \geq n_1 \wedge (f(\bar{0}))_j = c_2(g^{-1}(j))\big) \rightarrow \\
& (\forall i \in n_2)(\forall j \in n)\big(i \neq 0 \wedge j \geq n_1 \wedge \mathsf{Symbol}_{\Rightarrow}(\pi_1((f_i)_j)) \rightarrow \\
& \qquad \pi_2((f_{i^+})_j) = \pi_2((f_i)_j)^+ \wedge \pi_3((f_{i^+})_j) = \pi_3((f_i)_j)^+\big) \wedge \\
& (\forall i \in n_2)(\forall j \in n)\big(i \neq 0 \wedge j \geq n_1 \wedge \mathsf{Symbol}_{=}(\pi_1((f_i)_j)) \rightarrow \\
& \qquad \pi_2((f_{i^+})_j) = \pi_2((f_i)_j)^+ \wedge \pi_3((f_{i^+})_j) = \pi_3((f_i)_j)^+\big) \wedge \\
& (\forall i \in n_2)(\forall j \in n)\big(i \neq 0 \wedge j \geq n_1 \wedge \mathsf{Symbol}_{\forall}(\pi_1((f_i)_j)) \rightarrow \\
& \qquad \pi_3((f_{i^+})_j) = \pi_3((f_i)_j)^+\big) \qquad\qquad \Big).
\end{aligned}
$$

PROOF. The formula is plainly Σ_1 with a $(\forall\exists)_0$-matrix. The function g computes the sum of the domains of the two input codes c_1 and c_2. The function f maps the domain of c_2 to codes. In each one of the codes, the formula $(\forall i \in n_2)(\forall j \in n_1)(\mathsf{Cod}(f_i) \wedge \mathsf{dom}_C f_i = n \wedge (f_i)_j = (c_1)_i)$ guarantees that the first part of the code is exactly c_1. The formula $(\forall i \in n_2)(\forall j \in n)\big(j \geq n_1 \wedge (f(\bar{0}))_j = c_2(g^{-1}(j))\big)$ guarantees that the first element of f is the concatenation of the two codes without pointer correction. Note that, as g is injective, g^{-1} is easily characterized by an $(\exists)_0$-formula similarly to the direct image. The remaining formulae enforce that, at each i, all the pointers in the c_2 part of f_i are increased by 1 with respect to f_{i-1}, thus yielding a properly shifted code at f_{n_1}. □

As usual, we will write $\varphi_{\mathtt{cat}}(c_1, c_2, d, w)$ to indicate the same formula without the existential quantifiers, but where w is a quintuple whose components are f, g, n_1, n_2 and n.

LEMMA 9
The predicate $\mathsf{Subst}(c, t, d)$ that holds when the formula with code d results from the formula with code c by replacement of the first variable with the code of a term t is strongly represented by the following Σ_1-formula $\varphi_{\mathtt{sub}}(c, t, d)$, whose matrix is of form $(\forall\exists)_0$:

$$
\begin{aligned}
&\exists m \exists f \exists c' \exists w' \exists f' \exists n \exists n' \exists k \exists q \\
\Big(\;& (\exists i \in \mathsf{dom}_C\, c)\big(\mathsf{VarIn}(m, c_i)\big) \wedge (\forall i \in \mathsf{dom}_C\, c)(c_i = d_i) \wedge \\
& (\forall i \in \mathsf{dom}_C\, C)(\forall v \in \pi_2(\mathsf{ran}_C\, c))(\mathsf{VarIn}(v, c_i) \rightarrow m \leq v) \wedge \\
& (\forall i \in \mathsf{dom}_C\, C)(\forall v \in \pi_3(\mathsf{ran}_C\, c))(\mathsf{VarIn}(v, c_i) \rightarrow m \leq v) \wedge \\
& \mathsf{Cod}(c') \wedge \varphi_{\mathsf{cat}}(c, q, c', w') \wedge \mathsf{dom}_C\, d = \mathsf{dom}_C\, c' + 2 \wedge \\
& n = \mathsf{dom}_C\, c \wedge n' = \mathsf{dom}_C\, c' \wedge \varphi_{\mathtt{ind}}(t, k, f') \wedge \varphi_{\mathtt{ithc}}(m, k, q) \wedge \\
& (\forall i \in n')(d_i = c'_i) \wedge \\
& \mathsf{Symbol}_{\Rightarrow}(\pi_1(d_{n'})) \wedge \pi_2(d_{n'}) = n'^{-} \wedge \pi_3(d_{n'}) = n^{-} \wedge \\
& \mathsf{Symbol}_{\forall}(\pi_1(d_{n'^+})) \wedge \pi_2(d_{n'^+}) = m \wedge \pi_3(d_{n'^+}) = n' \qquad\qquad \Big).
\end{aligned}
$$

Here, $\mathsf{dom}_C\, c' + 2$ is a shorthand for $\big((\mathsf{dom}_C\, c')^+\big)^+$ and $\varphi_{\mathsf{ithc}}(m, k, q)$ is a $(\forall\exists)_0$-formula that holds when q is the code of the formula $m = \overline{(k)}_C$ defined in the obvious (albeit longish) way. The predicate $\mathsf{VarIn}(i, t)$ checks whether the variable v_i is present in the triple t and is not bound

(implicitly via bound lists). Its definition is plain, being similar to the one of PtBy, and it clearly is $(\exists)_0$.

More specifically, we have that for each formula φ and term code t:

$$T' \vdash \mathsf{Subst}(\ulcorner\varphi\urcorner, t, \ulcorner\forall x(x = t \to \varphi)\urcorner)$$
$$T' \vdash \forall z(\mathsf{Subst}(\ulcorner\varphi\urcorner, t, d) \to z = \ulcorner\forall x(x = t \to \varphi)\urcorner).$$

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.

PROOF.The statement is trivially implied by the conditions in $\varphi_{\mathtt{sub}}$ thanks to the previous lemma and fact, as they force d to be the code of $\ulcorner\forall x(x = t \to \varphi)\urcorner$ when $c = \ulcorner\varphi\urcorner$. □

LEMMA 10

The function which evaluates all the renamings in a formula is strongly representable through a Σ_1-formula with a $\forall\exists$-matrix $\varphi_{\mathtt{ren}}(c, d)$, namely

$$\begin{aligned}
\exists f \exists l\{\ & \mathsf{Fnc}(f) \wedge \mathsf{Cod}(c) \wedge \mathsf{Cod}(d) \wedge \mathsf{dom}\, f = \mathsf{dom}_C\, c \wedge \\
& l^+ = \mathsf{dom}\, f \wedge f(\overline{0}) = \langle\langle l, \overline{0}, \overline{0}\mathsf{rangle}, c\mathsf{rangle} \wedge d = f_l \wedge \\
& (\forall i \in \mathsf{dom}\, f)[\ \mathsf{Cod}(\pi_2(f_{(i+1)})) \wedge \mathsf{dom}\, f_{i+1} = \mathsf{dom}_C\, c \wedge \\
& \quad \mathsf{Fnc}(\pi_1(f_{i+1}) \wedge \mathsf{Num}(\mathsf{dom}\, \pi_1(f_{i+1})) \wedge \\
& \quad \mathsf{dom}_C\, \pi_2(f_{i+1}) = \mathsf{dom}_C\, c \wedge \pi_2(f_{i+1})(\overline{0}) = c(\overline{0}) \wedge \\
& \quad (\forall j \in \mathsf{dom}_C\, c)(\ j \notin \pi_1^3(\pi_1(f_i)) \to \\
& \qquad \pi_2(f_i)(j) = \pi_2(f_{i+1})(j)) \\
& \quad (\forall j \in \mathsf{dom}_C\, c)(\forall z \in \mathsf{dom}\, \pi_1(f_i))(j = \pi_1^3(\pi_1(f_i)(z)) \to (\star)) \wedge \\
& \quad (\forall j \in \mathsf{dom}\, \pi_1(f_{i+1}))(\#)]\}
\end{aligned}$$

— apex 2 omitted in all pair projections —, where $(\star)$ is the formula

$$\begin{aligned}
& (\forall k \in c(\overline{0}))(\forall h \in c(\overline{0}))(\forall s \in c(\overline{0}))(\forall v_1 \in c(\overline{0}))(\forall v_2 \in c(\overline{0})) \\
& (s = \pi_1^3(\pi_2(f_i)) \wedge h = \pi_2^3(\pi_2(f_i)) \wedge k = \pi_3^3(\pi_2(f_i)) \wedge \\
& v_1 = \pi_2^3(\pi_1(f_i)(z)) \wedge v_2 = \pi_3^3(\pi_1(f_i)(z)) \to (\star)_2),
\end{aligned}$$

$(\star)_2$ is the conjunction of the following implications:

$$\begin{aligned}
& \mathsf{Symbol}_{\Rightarrow}(s) \to \langle h, v_1, v_2\rangle \in \pi_1(f_{i+1}) \wedge \langle k, v_1, v_2\rangle \in \pi_1(f_{i+1}), \\
& \mathsf{Symbol}_{=}(s) \wedge h = v_1 \wedge k \neq v_1 \to \langle s, v_2, k\rangle = \pi_2(f_{i+1})(j), \\
& \mathsf{Symbol}_{=}(s) \wedge k = v_1 \wedge h \neq v_1 \to \langle s, h, v_2\rangle = \pi_2(f_{i+1})(j), \\
& \mathsf{Symbol}_{=}(s) \wedge k = v_1 \wedge h = v_1 \to \langle s, v_2, v_2\rangle = \pi_2(f_{i+1})(j), \\
& \mathsf{Symbol}_{\in}(s) \wedge h = v_1 \wedge k \neq v_1 \to \langle s, v_2, k\rangle = \pi_2(f_{i+1})(j), \\
& \mathsf{Symbol}_{\in}(s) \wedge k = v_1 \wedge h \neq v_1 \to \langle s, h, v_2\rangle = \pi_2(f_{i+1})(j), \\
& \mathsf{Symbol}_{\in}(s) \wedge k = v_1 \wedge h = v_1 \to \langle s, v_2, v_2\rangle = \pi_2(f_{i+1})(j), \\
& \mathsf{Symbol}_{R_q}(s) \wedge k = v_1 \to \pi_2(f_i)(k) = \pi_2(f_{i+1})(j) \wedge \langle j, q, v_2\rangle \in \pi_1(f_{i+1}), \\
& \mathsf{Symbol}_{R_q}(s) \wedge k \neq v_1 \to \pi_2(f_i)(k) = \pi_2(f_{i+1})(j) \wedge \langle j, q, k\rangle \in \pi_1(f_{i+1}), \\
& \mathsf{Symbol}_{\forall}(s) \wedge h \neq v_1 \to \langle k, v_1, v_2\rangle \in \pi_1(f_{i+1}) \wedge \langle s, h, k\rangle = \pi_2(f_{i+1})(j), \\
& \mathsf{Symbol}_{\forall}(s) \wedge h = v_1 \to \langle s, h, k\rangle = \pi_2(f_{i+1})(j)
\end{aligned}$$

and (#) states that $\pi_1(f_{i+1})$ has only the triples that $(\star)$ enforces. The latter formula, which is similar to $(\star)$, is $(\forall\exists)_0$ and it states that if a node is present in $\pi_1(f_{i+1})$, then one of the cases in $(\star)$ holds in f_i.

(Note that we have resorted to slight abuses of notation for the sake of clarity, e.g. $j \notin \pi_1(\pi_1(f_i))$ means $(\forall z \in \mathsf{dom}\pi_1(f_i))(j \neq \pi_1(\pi_1(f_i))(z))$ as $\pi_1(f_i)$ is a function whose elements are triples).

PROOF. The idea is that f is a sequence of pairs in $\left(\bigcup_{i\in\mathbb{N}}(\mathbb{N}^3)^i\right) \times \mathbf{Cod}$ (where **Cod** is the class of all codes), which represents the state of an iterative renaming algorithm. At each step, the second projection is the code at that iteration, and the first projection is a 'to-do list'. Indeed, the first element of the pair is a list of triples whose first element indicates an index in the code (a subformula) to check for renaming, the second is the index of the variable that has to be renamed and the third the variable it has to be renamed to. In the initial state, the second projection is the input code and the list of triples contains the triple $\langle l, \overline{0}, \overline{0}\rangle$, indicating to do an empty renaming starting from the root of the formula (index l).

After the ith iteration, thanks to the formulae ($\star$) and (#), the ith level of the syntax tree of the formula is processed: all the variables in the level are checked against the current to-do list, and every renaming symbol in the level pushes new a renaming in its subformula for the next iteration.

Let i be the current iteration. In each iteration, all the subformulae of the code are checked against all the entries in the to do list; let j be the index of some subformula $\langle s, h, k\rangle$ of the current code (s is the symbol at the root of the subformula, h its left subformula, k its right subformula) such that the triple $\langle j, v_1, v_2\rangle$ is in the to-do list (in the formula, such triple has index z in the list). It is clear how the cases for $\Rightarrow$, $\forall$ and $\in$ in the formula ($\star$) encode the application of the renaming in the given triple by specifying how the subformula becomes at the step $i + 1$ in the several cases and by propagating renamings to its subformulae.

It remains to check what happens in the cases in which a renaming operator R_q is analysed. Renaming operators substitute themselves with their child and push a new triple indicating that the child has to be analysed with the substitution $q \mapsto k$. In the case in which k should be renamed to some variable v_2, the pushed triple contains the substitution $q \mapsto v_2$.

From the previous discussion, $|\mathsf{dom}_C\, c|$ iterations are always sufficient to apply all renamings to the starting code. □

LEMMA 11 (Fixpoint).
Given a formula $\varphi(c)$, there is a formula χ that has the same free variables as φ, save c, such that $T' \vdash \chi \leftrightarrow \forall c\big(c = \ulcorner \chi \urcorner \to \varphi(c)\big)$, 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, $\forall c\big(c = \ulcorner \psi \urcorner \to \exists d, w D(c, d, w)\big)$ and $\forall c\big(c = \ulcorner \psi \urcorner \to \forall d, w\big(D(c, d, w) \to d = \ulcorner \forall c(c = \ulcorner \psi \urcorner \to \varphi)\urcorner\big)\big)$. Putting $\psi = \forall d, w(D(c, d, w) \to \varphi(d))$, we have $\forall c\big(c = \ulcorner \forall d, w(D(c, d, w) \to \varphi(d))\urcorner \to \forall d, w\big(D(c, d, w) \to d = \ulcorner \forall c(c = \ulcorner \forall d, w(D(c, d, w) \to \varphi(d))\urcorner \to \varphi)\urcorner\big)\big)$. Let $\chi = \forall c(c = \ulcorner \forall d, w(D(c, d, w) \to \varphi(d))\urcorner \to \forall d, w(D(c, d, w) \to \varphi(d)))$. Then $\forall c \forall d \forall w\big(c = \ulcorner \forall d, w(D(c, d, w) \to \varphi(d))\urcorner \wedge D(c, d, w) \to d = \ulcorner \chi \urcorner\big)$. Since $\forall c\big(c = \ulcorner \psi \urcorner \to \exists d, w D(c, d, w)\big)$, we have $\exists d, w\big(c = \ulcorner \forall d, w(D(c, d, w) \to \varphi(d))\urcorner \wedge D(c, d, w)\big)$ for any c such that $c = \ulcorner \psi \urcorner$. Thus, as c, d, w are not free in φ, we have

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

□

In general, $c = \overline{(k)}_C$ is a Σ_1-formula with a $(\forall\exists)_0$-matrix, while 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, while 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.

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

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, 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, $\mathbf{C}(x) = \exists z,u,y,w\big(z = \ulcorner\varphi_C\urcorner \wedge \varphi_{\mathtt{sub}}(z,x,u,w) \wedge \mathsf{Proof}(u,y)\big)$ is a 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 12
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 the following corollary.

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 Related Work

In [18], a proof is provided of essential undecidability with respect to the class of $(\forall\exists\forall)_0$-formulae under an axiomatic core closely related to the one adopted hereinabove (we have only added a few

instances of **Sep**). The main differences originating from our reduction in the number of quantifier alternations are the following.

1. In [18], the definition of the Num property is more straightforward, as the second alternation of quantifiers allows one to directly characterize successors, with no need of separate functions 'storing' the needed successors. The idea of using the successor function to reduce the Δ_0-complexity of the natural number predicate comes from [4], where the first limit ordinal and the successor function on natural numbers are characterized together by means of a $(\forall\exists)_0$-formula. We imported the idea of this simultaneous definition by re-adapting it to finite ordinals (as no guarantee of existence of an infinite set is given in a very weak axiomatic core).
2. In [18], the definition of code is more sophisticated due to one more inner universal quantification. In the present work, codes are simplified to the bare minimum in order to express them and a total order on them by $(\forall\exists)_0$-formulae. The present definition lacks some desirable properties, which makes most of the subsequent definitions unavoidably more involved. In [18], for instance, checking whether or not two codes represent the same formula is easily done by just checking whether their last components (representing the roots of the respective trees) are the same. Our sameness comparison between codes is more roundabout, as our encoding does not embed children syntax trees inside their parents, but just represents them via *pointers*, i.e. indices in the code-sequence.[9] Checking whether two codes represent the same formula hence becomes a much more complicated endeavour, for which we require the codes to be in some sort of normal form (using the *CleanCopy* predicates defined in p.55). In general, the difference in the straightforwardness of the code definition, which by itself paves the way to a reduction in the Δ_0-complexity, is responsible for all further differences in the definition of the Proof predicate.
3. The technique outlined in Remark 1 is used throughout the paper to reduce the complexity of almost every specification.

Much as in [18], our aim has been to keep the complexity of the arithmetization machinery low—a new goal with respect to [24], as recalled in the Introduction.

7 Conclusions

The claim of Corollary 2 is closely akin to the content of [4, Sec. 2], but the framework in which this paper has cast it 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 [18], 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 [17]—can improve this tradeoff.

Meta-level reasoning is often exploited in multi-decision making or collaborative autonomous agent systems. We hope that a theoretical investigation such as the one developed above can also be clarifying in the practice of agent-based AI, but it goes without saying that there is an unavoidable

[9] One can find in [25] an interesting discussion on the double option 'value-based vs pointer-based' representation of binary trees in a framework akin to ours.

gap between a study concerning the first principles and the pragmatic needs of any software application.

Acknowledgements

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.

References

[1] D. Cantone, M. Nicolosi Asmundo, C. Chiaruttini and E. G. Omodeo. Cumulative hierarchies and computability over universes of sets. *Le Matematiche*, **63**, 31–84, 2008.

[2] D. Cantone, E. G. Omodeo and A. Policriti. *Set Theory for Computing: From Decision Procedures to Declarative Programming with Sets*. Texts and Monographs in Computer Science. Springer, 2001.

[3] D. Cantone, V. Cutello and A. Policriti. Set-theoretic reductions of Hilbert’s tenth problem. In *CSL ‘89, 3rd Workshop on Computer Science Logic*, Kaiserslautern, Germany, 2–6 October 1989, E. Börger, H. K. Büning and M. M. Richter, eds. Vol. 440 of *Lecture Notes in Computer Science*, pp. 65–75. Springer, 1990.

[4] D. Cantone, E. G. Omodeo and M. Panettiere. From Hilbert’s $10^t h$ problem to slim, undecidable fragments of set theory. In *Proc. of the 21st Italian Conference on Theoretical Computer Science, ICTCS 2020*, G. Cordasco, L. Gargano and A. A. Rescigno, eds. Vol. 2756 of *CEUR Workshop Proc.*, pp. 47–60. CEUR-WS.org, 2020.

[5] D. Cantone, E. G. Omodeo and M. Panettiere. Very weak, essentially undecidabile set theories. In *Proceedings of the 36th Italian Conference on Computational Logic*, Parma, Italy, 7–9 September 2021, S. Monica and F. Bergenti, eds. Vol. 3002 of *CEUR Workshop Proceedings*, pp. 31–46. CEUR-WS.org, 2021.

[6] S. Costantini and V. Pitoni. Towards a logic of “Inferable” for self-aware transparent logical agents. In *Proc. of Italian Workshop on Explainable Artificial Intelligence, XAI.it@AIxIA 2020*, Online Event, 25–26 November 2020, C. Musto, D. Magazzeni, S. Ruggieri and G. Semeraro, eds. Vol. 2742 of *CEUR Workshop Proc.*, pp. 68–79. CEUR-WS.org, 2020.

[7] A. Formisano, E. Omodeo and A. Policriti. Reasoning on relations, modalities, and sets. In *Ewa Orłowska on Relational Methods in Logic and Computer Science*, J. Golińska-Pilarek and M. Zawidzki, eds, pp. 129–168. Springer International Publishing, 2018.

[8] A. Formisano, E. G. Omodeo and A. Policriti. Three-variable statements of set-pairing. *Theoretical Computer Science*, **322**, 147–173, 2004.

[9] J. van Heijenoort. From Frege to Gödel—A source book in mathematical logic, 1879–1931. In *Source Books in the History of the Sciences*, 3rd edn. Harvard University Press, 1977.

[10] P. M. Hill and J. W. Lloyd. *The Gödel Programming Language*. MIT Press, 1994.

[11] T. Jech. *Set Theory*, Third Millennium edn. Springer Monographs in Mathematics. Springer, Berlin Heidelberg, 2003.

[12] A. Levy. *A Hierarchy of Formulas in Set Theory*. Vol. 57. American Mathematical Society, Providence, RI, 1965.

[13] E. Mendelson. *Introduction to Mathematical Logic*. Chapman & Hall, 2015.

[14] R. Montague. Semantical closure and non-finite axiomatizability I. In *Infinitistic Methods. Proc. of the Symposium on Foundations of Mathematics*, Warsaw, 2–9 September 1959, pp. 45–69. Państwowe Wydawnictwo Naukowe, Warsaw, and Pergamon Press, Oxford–London–New York–Paris, 1961.
[15] J. von Neumann. Zur Einführung der trasfiniten Zahlen. *Acta Scientiarum Mathematicarum*, **1**, 199–208, 1922. Available in English translation in [9, pp. 346–354].
[16] M. Panettiere. *Essential Undecidability: Foundations versus Proof Technology*. Master's Thesis, Università degli Studi di Catania, Italy, 2021.
[17] F. Parlamento and A. Policriti. The logically simplest form of the infinity axiom. *Proceedings of the American Mathematical Society*, **103**, 274–276, 1988.
[18] F. Parlamento and A. Policriti. Decision procedures for elementary sublanguages of set theory. IX. Unsolvability of the decision problem for a restricted subclass of the $\{\Delta\}_0$-formulas in set theory. *Communications on Pure and Applied Mathematics*, **41**, 221–251, 1988.
[19] F. Parlamento and A. Policriti. Undecidability results for restricted universally quantified formulae of set theory. *Communications on Pure and Applied Mathematics*, **46**, 57–73, 1993.
[20] R. M. Robinson. The theory of classes, a modification of von Neumann's system. *J. Symb. Logic*, **2**, 29–36, 1937.
[21] R. M. Robinson. An essentially undecidable axiom system. In *Proc. of the International Congress of Mathematicians* (Harvard University, Cambridge, MA, 30 August to 6 September 1950). Vol. 1, pp. 729–730. AMS, Providence, RI, 1952.
[22] A. Tarski. Sur les ensembles fini. *Fundamenta Mathematicae VI*, **6**, 45–95, 1924.
[23] A. Tarski and S. Givant. *A Formalization of Set Theory without Variables*. Vol. 41 of Colloquium Publications. American Mathematical Society, 1987.
[24] R. L. Vaught. On a theorem of Cobham concerning undecidable theories. In *Proc. of the 1960 International Congress on Logic, Methodology, and Philosophy of Science*, E. Nagel, P. Suppes and A. Tarski, eds. pp. 14–25. Stanford University Press, 1962.
[25] G. Weiss. *Recursive Data Types in Setl: Automatic Determination, Data Language Description, and Efficient Implementation (Compilers)*. PhD Thesis, New York University, USA, 1986.

Received 25 May 2022