INFINITE TIME COMPUTABLE MODEL THEORY

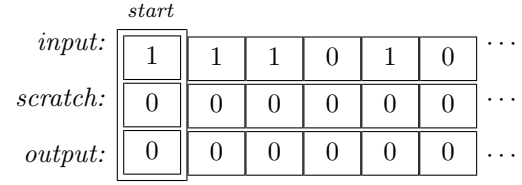
JOEL DAVID HAMKINS, RUSSELL MILLER, DANIEL SEABOLD, AND STEVE WARNER

Abstract. We introduce infinite time computable model theory, the computable model theory arising with infinite time Turing machines, which provide infinitary notions of computability for structures built on the reals $\mathbb{R}$. Much of the finite time theory generalizes to the infinite time context, but several fundamental questions, including the infinite time computable analogue of the Completeness Theorem, turn out to be independent of ZFC.

1. Introduction

Computable model theory is model theory with a view to the computability of the structures and theories that arise (for a standard reference, see [EGNR98]). Infinite time computable model theory, which we introduce here, carries out this program with the infinitary notions of computability provided by infinite time Turing machines. The motivation for a broader context is that, while finite time computable model theory is necessarily limited to countable models and theories, the infinitary context naturally allows for uncountable models and theories, while retaining the computational nature of the undertaking. Many constructions generalize from finite time computable model theory, with structures built on $\mathbb{N}$, to the infinitary theory, with structures built on $\mathbb{R}$. In this article, we introduce the basic theory and consider the infinitary analogues of the completeness theorem, the Löwenheim-Skolem Theorem, Myhill's theorem and others. It turns out that, when stated in their fully general infinitary forms, several of these fundamental questions are independent of ZFC. The analysis makes use of techniques both from computability theory and set theory. This article follows up [Ham05].

1.1. **Infinite time Turing machines.** The definitive introduction to infinite time Turing machines appears in [HL00], but let us quickly describe how they work. The



hardware of an infinite time Turing machine is identical to a classical (three tape) Turing machine, with a head reading and writing 0s and 1s on the one-way infinite tapes, following the instructions of a finite program with finitely many states.

MSC: 03D60; 03D45; 03C57; 03E15. Keywords: infinite time Turing machines, computable model theory. The research of the first two authors has been supported in part by grants from the Research Foundation of CUNY, and the first author is additionally thankful to the Institute for Logic, Language and Computation and the NWO (Bezoekersbeurs B62-612) for supporting his summer 2005 stay at Universiteit van Amsterdam.

Computation begins with the input on the *input* tape and the head on the left-most cell in the *start* state. Successor steps of computation are determined by the program in exactly the classical manner. At any limit ordinal stage, as a matter of definition, the machine resets the head to the left-most cell, assumes the *limit* state and updates the tape so that every cell exhibits the lim sup of the previous values displayed in that cell. This is equivalent to using the limit value, if the value displayed by the cell has stabilized, and otherwise 1. Computation ceases only when the *halt* state is explicitly obtained, and in this case the output is whatever is written on the output tape. (If the head falls off the tape, no output is given.) If p is a program, it computes a function φ_p, defined by $\varphi_p(x) = y$ if and only if on input x the computation determined by p leads to output y. The natural context here for input and output is the Cantor space ${}^{\omega}2$ of all infinite binary sequences, which we will denote by $\mathbb{R}$ and refer to as the set of reals. A (partial) function $f : \mathbb{R} \to \mathbb{R}$ is infinite time *computable* if it is φ_p for some program p. Binary and n-ary functions can be equivalently modelled either by adding additional input tapes, or by viewing a single real as the interleaving of the digits of n many reals. A set $A \subseteq \mathbb{R}$ is infinite time *decidable* if its characteristic function is infinite time computable. The set A is infinite time *semi-decidable* if the function $1 \restriction A$ with domain A and constant value 1 is computable. In this article, we will freely use the terms *computable* and *decidable* to mean infinite time computable and infinite time decidable, though we will sometimes specify "infinite time" for clarity. When referring to the classical notions of computability, we will always say "finite time computable" and "finite time decidable." We regard the natural numbers $\mathbb{N}$ as coded in $\mathbb{R}$ by identifying n with the binary sequence consisting of n ones followed by zeros. A real is *writable* if it is $\varphi_p(0)$ for some program p. A real is *accidentally* writable if it appears on one of the tapes during any computation $\varphi_p(0)$. A real is *eventually* writable if it appears on the output tape of a (not necessarily halting) computation $\varphi_p(0)$, and from some point on in that computation, it is never changed. An ordinal α is *clockable* if there is a computation $\varphi_p(0)$ moving to the *halt* state exactly on the α^{th} computational step.

The growing body of literature on infinite time Turing machines includes [HL00], [Wel00b], [Wel00a], [HS01], [LÖ1], [HL02], [Ham02], [HW03], [DHS05], [Ham05], [Wel05], [Koe05].

1.2. **Basic definitions.** The main idea will be that a computable model is one whose underlying set is decidable and whose functions and relations are uniformly computable. In order to make this precise, let us first be more specific about our syntax and how it is represented. A language consists of a collection of function, relation and constant symbols, with each function and relation symbol assigned a finite arity. In addition, every language has the logical connective symbols $\wedge$, $\vee$, $\neg$, $\rightarrow$, $\leftrightarrow$, parentheses, the equality symbol $=$, quantifiers $\forall$, $\exists$, variable symbols v_0, v_1, and so on. In finite time computable model theory, in order to bring these syntactic objects into the realm of computability, one views each symbol in the (countable) language as being represented by a particular natural number, its Gödel code, so that the various syntactic objects—such as terms, formulas and sentences—are simply finite sequences of these codes, which can in turn be coded with a single natural number.

Infinite time computable model theory, however, offers the possibility of *uncountable* computable models. And because we will want to consider the elementary or

atomic diagrams of such models, the possibility of uncountable languages is unavoidable. Clearly, we cannot expect to code such languages using Gödel codes only in $\mathbb{N}$. Therefore, we work in a more general context, where the symbols of a language are represented with Gödel codes in $\mathbb{R}$, rather than $\mathbb{N}$. This conforms with the philosophy of infinite time computability, where the fundamental inputs and outputs of computations are real numbers. A *computable presentation* of a language $\mathcal{L}$ is the assignment of a Gödel code $\ulcorner s \urcorner$ to every function, relation and constant symbol s in the language, in such a way that the set of such codes for symbols in $\mathcal{L}$ is decidable, and there are computable functions telling us, given any $\ulcorner s \urcorner$, what kind of symbol s is and, when it is a function or relation symbol, what arity it has. We assume that the basic logical symbols (logical connectives, = symbol, parentheses, variable symbols, quantifiers) have simple Gödel codes in $\mathbb{N}$.

Given the Gödel codes of the underlying symbols, one develops the Gödel coding of all the usual syntactic notions. For example, a term τ is a particular kind of finite sequence of function, constant and variable symbols, and we may assign the Gödel code $\ulcorner \tau \urcorner$ via the usual manner of coding finite sequences of reals with reals. Similarly, any formula φ in the language is a finite sequence of symbols from the language, and we can assign it a natural Gödel code. We assume that the Gödel coding of the language is undertaken in such a way that we can unambiguously determine whether a given Gödel code is the code of a formula or an individual symbol, and what kind; that from the Gödel code of a formula or term we can compute the Gödel codes of the subformulas and subterms; and that the Gödel codes are uniquely readable. For any computable presentation $\mathcal{L}$, it follows that all the elementary syntactic notions are computable from the Gödel codes, such as finding the inductive construction history of a formula or term or determining whether a given occurrence of a variable in a formula is free or not.

Definition 1. In the infinite time context, a *computable model* is a structure $\mathcal{A} = \langle A, f^{\mathcal{A}}, R^{\mathcal{A}}, c^{\mathcal{A}} \rangle_{f,R,c\in\mathcal{L}}$ in a language $\mathcal{L}$, with a fixed computable presentation of $\mathcal{L}$, such that the underlying set $A \subseteq \mathbb{R}$ of the model is decidable and the functions, relations and constants of $\mathcal{A}$ are uniformly computable from their input and the Gödel codes of their symbols. A structure has a *computable presentation* if it is isomorphic to a computable model.

A simple recursive argument shows that the value of any term $\tau(\vec{a})$ is uniformly computable from its Gödel code $\ulcorner \tau \urcorner$ and the input $\vec{a}$. It follows that one can compute the truth in $\mathcal{A}$ of any given atomic formula. Specifically, the atomic diagram of $\mathcal{A}$ is the set $\Delta_0(\mathcal{A}) = \{\, \varphi[\vec{a}] \mid \varphi \text{ atomic}, \vec{a} \in A^{<\omega}, \mathcal{A} \models \varphi[\vec{a}] \,\}$, and if $\mathcal{A}$ is a computable model, then we can decide, on input $\ulcorner \varphi \urcorner$ and $\vec{a}$, whether $\varphi[\vec{a}] \in \Delta_0(\mathcal{A})$. More generally, we define:

Definition 2. A model $\mathcal{A}$ is *(infinite time) decidable* if the full elementary diagram of the structure $\Delta(\mathcal{A}) = \{\, \varphi[\vec{a}] \mid \mathcal{A} \models \varphi[\vec{a}] \,\}$ is infinite time decidable.

We caution the reader that in the infinite time context, a decidable model might not be computable (see Corollary 10). This is a consequence of the phenomenon in infinite time computability that a function can have a decidable graph without being a computable function. The classical algorithm to compute a function from its graph relies on having an effective enumeration of the possible values of the function, but in the infinite time context we have no effective method to enumerate $\mathbb{R}$. For a purely relational model, with no function or constant symbols in the

language, however, this phenomenon is avoided and the model is computable if and only if its atomic diagram is decidable.

Another departure from the classical theory is that every computable model $\mathcal{A}$ with underlying set contained in $\mathbb{N}$ is decidable. The point is that the infinite time algorithm can systematically check the truth of any first order statement φ in $\mathcal{A}$, given the Gödel code $\ulcorner\varphi\urcorner$, by inductively applying the Tarski definition of truth. If φ has the form $\exists x\, \psi(x)$, then the algorithm simply checks the truth of all $\psi(n)$ for $n \in A$. More generally, if an infinite time Turing machine has the capacity for a complete search through the domain of a structure—for example if the domain consisted of a writable set of writable reals—then we will be able effectively to carry out the Tarski definition of truth. So one might want to regard such a situation as a special or trivial case in infinite time computable model theory. We refer to such a structure as a *writable structure*; a formal definition appears on page 11.

A theory (meaning any set of sentences in a fixed language) is *computably axiomatizable* if there is a theory T_0, having the same consequences as T, such that the set of Gödel codes $\{\, \ulcorner\varphi\urcorner \mid \varphi \in T_0 \,\}$ is decidable. A theory T is *decidable* if the set of Gödel codes of its consequences $\{\, \ulcorner\varphi\urcorner \mid T \vdash \varphi \,\}$ is decidable. If the underlying language is coded in $\mathbb{N}$, then every computably axiomatizable theory is decidable, because an infinite time algorithm is easily able to search through all proofs. More generally, if an algorithm can write a real listing all the Gödel codes of symbols in the language, then it can systematically generate the Gödel codes of all sentences in that language, determine which are axioms in T_0, and then generate a list of all possible proofs. This shows that any theory with a writable set of axioms has a writable set of theorems.

1.3. **Coding with reals.** We would like to view our algorithms as engaging with arbitrary countable objects, such as countable ordinals or theories, even though formally the machines treat only infinite binary sequences. So let us introduce a method of coding. We regard any real $x \in \mathbb{R}$ as coding a relation $\lhd$ on $\mathbb{N}$ by $i \lhd j$ if and only if the $\langle i, j\rangle^{\text{th}}$ bit of x is 1, using a bijective pairing function $\langle\cdot,\cdot\rangle$ on $\mathbb{N}$. For every countable ordinal α, there is such a relation $\lhd$ on $\mathbb{N}$ with $\langle \alpha, <\rangle \cong \langle A, \lhd\rangle$, where A is the field of $\lhd$. The set WO consists of the reals x coding such well ordered relations $\lhd$, and we refer to these as the reals coding ordinals. This is well known to be a complete Π^1_1 set of reals. One of the early results of [HL00] showing the power of infinite time Turing machines is that this set is decidable. We sketch the proof because the method will be useful for other purposes here.

Theorem 3. ([HL00, Theorem 2.2]) WO *is infinite time decidable.*

Proof. Given a real x, we first check whether x codes a linear order $\lhd$, by systematically checking all instances of transitivity, reflexivity and anti-symmetry, in ω many steps of computation. Assuming $\lhd$ is a linear order, we next attempt to find the $\lhd$-least element in the field of the relation. This can be done by placing a current guess for the least element on the scratch tape, and searching for a $\lhd$-smaller element. When such a better guess is found, the algorithm over-writes it on the scratch tape, and also flashes a special flag on and then off. At the next limit stage, if the flag is on, then the guess was changed infinitely many times, and so the real is rejected, because it does not code a well order. If the flag is off at a limit, then the guesses stabilized on the current $\lhd$-least element, which now appears on the scratch tape. Next, the algorithm erases all mention of this element from the field of

the relation coded on the input tape, and then continues to find (and subsequently erase) the next least element, and so on. The algorithm should detect limits of limit stages, so that the scratch tape and the flag can be accordingly reset. Eventually, the well ordered initial segment of $\lhd$ is erased from the field of the relation coded on the input tape. By detecting when the tape is empty, the algorithm can know whether the original real coded a well order. If not, the algorithm will detect the ill-founded part of it, and reject at that stage. □

Since WO is a complete Π^1_1 set, any Π^1_1 question reduces to a question about WO, and so we obtain:

Corollary 4. *Any Π^1_1 set is infinite time decidable. Hence, any Σ^1_1 set is also decidable.*

Any real x can be viewed as the code of an ω-sequence of reals $\langle (x)_n \mid n < \omega\rangle$ by $(x)_n(m) = x(\langle n, m\rangle)$. Thus, if we are also given a real z coding a relation $\lhd$ on $\mathbb{N}$ of order type α, then any $\beta < \alpha$ is represented by some n with respect to $\lhd$, and we may view x as coding via z an α-sequence $\langle x_\beta \mid \beta < \alpha\rangle$ of reals. The real x_β is $(x)_n$, where n is the β^{th} element with respect to $\lhd$.

More generally, any hereditarily countable set a can be coded with a real as follows. Suppose b is any countable transitive set containing a as an element, such as the transitive closure $\mathrm{TC}(\{a\})$, and let E be a relation on a subset $A \subseteq \mathbb{N}$ such that there is an isomorphism $\pi : \langle A, E\rangle \cong \langle b, \in\rangle$. Since this isomorphism π must be the Mostowski collapse of E, the set a is determined by E and the natural number n such that $\pi(n) = a$. We view the pair $\langle n, E\rangle$, coded by a real, as representing the set a. Of course, a set a generally has many different codes. In analogy with WO, let us define HC to be the set of such reals coding hereditarily countable sets in this way. Given two such codes x and y, define $x \equiv y$ if x and y are codes for the same set, and $x \in^* y$ if the set coded by x is an element of the set coded by y.

Theorem 5. *The structure $\langle \mathrm{HC}, \in^*, \equiv\rangle$ is infinite time computable.*

Proof. The elements of HC are precisely the reals coding pairs $\langle n, E\rangle$ where E is a well-founded relation on some $A \subseteq \mathbb{N}$, where A is the field of E, the natural number n is in A, and the structure $\langle A, E\rangle$ satisfies extensionality. Thus, the set HC is Π^1_1 definable, and hence decidable. The relation $x \equiv y$ is satisfied, where $x = \langle n, E\rangle$ and $y = \langle n', E'\rangle$ if and only if there is an isomorphism from the part of the field of E below n to the field of E' below n'. This is a Σ^1_1 property in the codes, and hence decidable. Similarly, the relation $x \in^* y$ simply asserts that there is some m in the field of E' such that $\langle n, E\rangle \equiv \langle m, E'\rangle$, which is also Σ^1_1, and hence decidable. □

The quotient structure $\mathrm{HC}/\equiv$, under the induced relation $\in^*$, is of course isomorphic to the transitive collection H_{ω_1} of hereditarily countable sets.

Theorem 6. *The satisfaction relation for hereditarily countable sets $\langle b, \in\rangle \models \varphi[\vec{a}]$ is infinite time decidable, given any code $\langle n, E\rangle \in \mathrm{HC}$ for b, the Gödel code $\ulcorner\varphi\urcorner$ and the code $\vec{n}$ of $\vec{a}$ with respect to E.*

Proof. This is simply an instance of the earlier remark we made, that when an algorithm has access to the entire domain of a structure, it can carry out the Tarski definition of truth. In this case, the code for b effectively provides the structure $\langle b, \in\rangle$ as a subset of $\mathbb{N}$. Alternatively, one could simply observe that the satisfaction relation has complexity Δ^1_1, and is therefore decidable. □

The constructible hierarchy of Gödel is the transfinite hierarchy of sets L_α, defined by: $L_0 = \emptyset$; $L_{\alpha+1}$ is the collection of definable subsets of L_α; for limit ordinals, $L_\eta = \bigcup_{\alpha<\eta} L_\alpha$. The constructible universe L is the proper class $\bigcup_\alpha L_\alpha$, and Gödel proved that $\langle L, \in\rangle$ is a (class) model of ZFC + GCH and much more.

Theorem 7.

(1) *There is an infinite time algorithm such that, on input a code for L_α for some countable ordinal α, writes a code of $L_{\alpha+1}$.*
(2) *There is an infinite time algorithm such that, on input a code of a countable ordinal α, writes a code of L_α.*

Proof. Given a code of L_α, one systematically considers each definition and each parameter, and by repeated applications of Theorem 6, one can write down codes for each of the definable subsets. This produces a code for $L_{\alpha+1}$. Given a code for α, one views $\mathbb{N}$ as an α-sequence of copies of $\mathbb{N}$. On each copy of $\mathbb{N}$, the algorithm may iteratively apply the previous method to produce codes for the successive new elements of L_β for each $\beta \leq \alpha$. □

The next theorem asserts that there is a real c such that an infinite time Turing machine can recognize whether a given real is c or not, but no algorithm can produce c on its own. This is like a person who is able to recognize a particular song, a lost melody, when someone else sings it, but who is unable to sing it on his or her own. The idea of the proof leads to the concept of L-codes for sets and ordinals, of which we will make extensive use later.

Lost Melody Theorem 8. ([HL00]) *There is a real c such that $\{c\}$ is infinite time decidable, but c is not writable.*

Proof. We sketch the proof from [HL00]. Results there show that every infinite time Turing machine computation either halts or repeats by some countable stage. Let β be the supremum of the stages by which all computations of the form $\varphi_p(0)$ have either halted or repeated. (Welch proved in [Wel00b] that $\beta = \Sigma$, the supremum of the accidentally writable ordinals.) The structure L_β is able to carry out all the computations $\varphi_p(0)$ for any length up to β, and so the defining property of β is expressible in L_β. One can use the defining property of β to show that there is a map from ω unbounded in β that is a definable subset of L_β. This map is therefore an element of $L_{\beta+1}$, and consequently β is countable in $L_{\beta+1}$. So there is some L-least real $c \in L_{\beta+1}$ coding a relation of order type β. This is the real we seek.

Notice that $\{c\}$ is decidable, because if we are given any candidate real c', we can check that it codes an ordinal β', and if so, we can write down a code for $L_{\beta'+1}$, and check whether $L_{\beta'+1}$ satisfies that β' is the supremum of the repeat points for all computations $\varphi_p(0)$. This will be true if and only if $\beta' = \beta$. Next, we check that c' is the least real in $L_{\beta'+1} = L_{\beta+1}$ coding $\beta' = \beta$. This will be true if and only if $c' = c$. So we can decide whether any given real is c or not.

Finally, c is not writable, because β is necessarily larger than every clockable ordinal, and hence larger than every writable ordinal. So β is not coded by any writable real. □

Corollary 9. *There is a function f that is not infinite time computable, but whose graph is infinite time decidable.*

Proof. Let $f(x) = c$ be the constant function with value c, the lost melody real. Since $\{c\}$ is decidable, we can decide the graph of f, which consists of all pairs (x, y) for which $y = c$. But f is not computable, since $c \neq \varphi_p(0)$ for every program p. □

Corollary 10. *There is an infinite time decidable model that is not infinite time computable.*

Proof. Let $\mathcal{A} = \langle \mathbb{R}, f \rangle$, where $f(x) = c$ is the constant function with value c, given by the Lost Melody Theorem, and $\ulcorner f \urcorner \in \mathbb{N}$. This is not a computable model, because the function f is not computable. Nevertheless, we will show that the elementary diagram of $\mathcal{A}$ is decidable. First, we consider the atomic diagram. We can use $f(f(x)) = f(x)$ to reduce the complexity of terms, and then observe that $f(x) = f(y)$ is always true and $f(x) = y$ amounts to $y = c$, which is decidable. So any atomic assertion is decidable. To decide the full elementary diagram, we observe that it admits the effective elimination of quantifiers down to Boolean combinations of assertions of the form $x = c$ and $x = y$ (plus true and false). The quantifier case essentially amounts to observing that $\exists x\,(x = c \ \ \& \ \ x \neq y)$ is equivalent to $y \neq c$ and $\exists x\,(x \neq c \ \ \& \ \ x \neq y)$ is simply true. So $\mathcal{A}$ is decidable, but not computable, concluding the proof.

This Corollary can also be proved by using a language with a single constant symbol 0, with $\ulcorner 0 \urcorner \in \mathbb{N}$. The structure $\mathcal{B} = \langle \mathbb{R}, c \rangle$, interpreting 0 as the Lost Melody real c, is not a computable model because the value of the constant is not computable from its Gödel code. But the structure $\mathcal{B}$ is simply an infinite model with a distinguished constant, which admits the elimination of quantifiers, and since one can decide all statements of the form $x = c$, it follows that $\mathcal{B}$ has a decidable theory. □

The idea of the Lost Melody Theorem provides a method of coding countable ordinals in L with unique codes. Specifically, for any $\alpha < \omega_1^L$, let β be least above α such that β is countable in $L_{\beta+1}$, and let c be the L-least real in $L_{\beta+1}$ coding a relation $\lhd$ on $\mathbb{N}$ with order type β. The ordinal α is represented by some natural number n with respect to $\lhd$, and so we will define $\langle n, c \rangle$ to be the *L-code* of α. Note that every ordinal α that is countable in L has exactly one L-code, since α determines β, which determines c, which determines $\lhd$, which determines n. Since the L-code of α is also a code of α in the sense of HC, we can computably determine by Theorem 5 whether $\alpha < \beta$, given L-codes for α and β. And just as with HC in this case, we can computably construct the isomorphism from the field of the relation coding α to the appropriate initial segment of the field of the relation coding β, and find the particular natural number representing α with respect to the code for β.

Lemma 11. *The set of L-codes for countable ordinals is infinite time decidable.*

Proof. Given a real coding a pair $\langle n, c \rangle$, we can determine whether c is the code of a relation $\lhd$ on $\mathbb{N}$ that is a well order of some order type β. If so, we can construct a code for $L_{\beta+1}$ and check that $L_{\beta+1}$ satisfies that β is countable and that the L-least real coding a relation of order type β is c. Finally, we can check that $L_{\beta+1}$ thinks that β is least such that it satisfies that α, the ordinal coded by n with respect to $\lhd$, is countable. If all these tests are passed, then the pair $\langle n, c \rangle$ is the L-code of α. □

More generally, we have L-codes for any set that is hereditarily countable in L. Specifically, suppose that a is any set that is hereditarily countable in L. Let β be least such that $a \in L_\beta$ and β is countable in $L_{\beta+1}$. It follows that L_β is countable in $L_{\beta+1}$, so there is some L-least real c coding a relation E such that $\langle \mathbb{N}, E\rangle \cong \langle L_\beta, \in\rangle$. The set a is represented by some natural number n with respect to E, and the *L-code* of a is the pair $\langle n, c\rangle$. Let LC be the set of such L-codes for hereditarily countable sets in L. Since these are also codes for sets in the sense of HC, it follows by Theorem 5 that we may computably decide the relation $\in^*$ on the codes induced by the $\in$ relation on the sets coded.

Theorem 12. *The structure $\langle L_{\omega_1^L}, \in\rangle$ has an infinite time computable presentation as $\langle \mathrm{LC}, \in^*\rangle$.*

Proof. The set $L_{\omega_1^L}$ is precisely HC^L, the sets that are hereditarily countable in L, and this is isomorphic to $\langle \mathrm{LC}, \in^*\rangle$ via the L-codes. The $\in^*$ relation is computable on the L-codes, just as in Theorem 5. And the set of L-codes LC is decidable just as in Lemma 11. □

Similarly, using the L-codes for ordinals, we see that the structure $\langle \omega_1^L, <\rangle$ has an infinite time computable presentation.

2. Arithmetic on the Real Line

As a straightforward example of an infinite time computable structure, we consider the most prominent uncountable structure in mathematics, the real line under arithmetic.

Lemma 13. *The standard structure $\mathcal{R}$ of the real line under addition, multiplication, subtraction, division, and the order relation $<$ is infinite time computably presentable.*

We use "the real line" to describe this structure, and refer to its elements as "points," because elsewhere in this paper we use the term "real number" to refer to elements of 2^ω. Also, since division by zero is usually undefined, let us regard it as a function on the computable domain $\mathcal{R} \times (\mathcal{R} - \{0\})$.

Proof. It is straightforward to identify points x on the real line uniquely with binary sequences $C \in 2^\omega$ such that $C(2n) = 0$ for infinitely many n and $C(2n+1) = 0$ for all but finitely many n and $C \neq \langle 1000\cdots\rangle$. The element C corresponds to the real point

$$(-1)^{C(0)} \left(\sum_{n=0}^{\infty} 2^n \cdot C(2n+1) + \sum_{n=1}^{\infty} \frac{C(2n)}{2^n} \right),$$

and C is called the *presentation* of this real point. (The condition $C \neq \langle 1000\cdots\rangle$ rules out the second presentation of the point 0 as -0.) The domain of our structure $\mathcal{R}$ is the set of all presentations of real points, and is decidable in infinite time, since each of the conditions can be checked in ω many steps. Also, it is easy to give a process for deciding (in infinite time) whether two given domain elements are equal, and if not, which is larger under $<$.

Of course, all of the usual arithmetic operations on these representations, such as sum, difference, product and quotient, have complexity (much less than) Δ^1_1 in the input, and therefore, by Corollary 4, these are all infinite time computable

operations. Nevertheless, for illustration let us show in moderate detail how to compute the sum C'' of two presentations C and C' of positive real points. First, in ω many steps, we find the greatest $k > 0$ such that $C(2k) = C'(2k)$, or else establish that there are infinitely many such k. Then we have two cases. If there is a greatest k, then beyond the k-th bit C and C' complement each other perfectly, and there are only finitely many bits left to add. Otherwise, there are infinitely many k with $C(2k) = C'(2k)$, and we build the sum from the inside, by always searching for the next greater bit k with $C(2k) = C'(2k)$ and computing C'' up to that bit. (The point is that when $C(2k) = C'(2k)$, we know right away whether we need to "carry" a 1 from $C''(2k)$ when calculating $C''(2k-2)$, even without knowing $C''(2k)$ itself yet.) This defines the entire sequence C''. Note that if the representation of the sum happens to have $C''(2k) = 1$ for a tail segment, then one must switch to the preferred representation by changing these bits to 0 and performing an additional carry.

Notice that each of the two cases could be carried out in finite time computability, producing each bit $C''(n)$ in finitely many steps, assuming that one was given oracles presenting C and C'. Infinite time is required only to decide which of the two cases to use, and (in the first case) to find the greatest k.

Addition of two negative real points can be defined using the above algorithm conjugated by the negation map $x \mapsto -x$, which is immediately seen to be computable. To get addition of a positive to a negative, we define subtraction of a positive real C' from another one $C > C'$, by taking finite approximations of the difference, adding them to C', and checking whether each finite approximation yields a sum $> C$ or $\leq C$.

It is tempting to bypass the discussion for subtraction by saying that the difference $C - C'$ should be that domain element D such that $C' + D = C$, since we have already given a method of computing the sum of positive domain elements. However, this does not suffice to prove computability, and indeed it illustrates a fundamental difference between the contexts of finite and infinite time: in infinite time computability, we may no longer have such effective search procedures. Without an infinite-time-computable enumeration of the domain of $\mathcal{R}$, there is no guarantee that we would ever find the element D described above, even though it must lie somewhere in the domain of $\mathcal{R}$. Therefore, it is necessary to compute D directly in infinite time, rather than searching for a D which satisfies $C' + D = C$. Decidability of subtraction as a ternary relation (that is, decidability of the statement $C - C' = D$) does follow from decidability of the addition relation, which follows from computability of addition as a function, but computability of subtraction is stronger.

For multiplication of positive domain elements C and C', we simply multiply C by each individual bit of C' (for instance, if $C'(2n) = 1$, then the product of C with that bit maps each bit of C n places to the right) and add the results together, one by one, in ω^2 many steps. Clearly each bit on the output tape does converge to a limit, since $C(2n+1) = 0$ for all but finitely many n, and the final output is the product of C and C'. This extends easily to the case of non-positive domain elements, so multiplication is computable. Finally, for division, we can check whether the divisor is the real point 0, and if not, we define it using the multiplication function, just as subtraction was defined using addition. Thus division is indeed a computable function on the domain $\mathcal{R} \times (\mathcal{R} - \{0\})$. □

One can expand the real field $\mathcal{R}$ to include all the usual functions of analysis: e^x, $\sqrt{x}$, $\ln x$, $\sin x$ and so on. Since (the bit values of) these functions have complexity below Δ^1_1, they are all infinite time computable by Corollary 4.

Let us turn now to the subfield $\mathcal{R}_w$, consisting of those real points having a writable presentation. It is clear from the algorithms given in the proof of Lemma 13 that $\mathcal{R}_w$ is a substructure of the real line $\mathcal{R}$. Moreover, we have the following lemma.

Lemma 14. *In the infinite time context, the ordered field $\mathcal{R}_w$ is computably presentable, and more generally, the ordered field $\mathcal{R}_w^X$ of those real points which have presentations writable using any oracle $X \subseteq \mathbb{R}$ is X-computably presentable. In each presentation, there is a computable (resp. X-computable) function from domain elements to the binary expansions of the real points they represent.*

Proof. The main difficulty is in getting the domain of our presentation of $\mathcal{R}_w$ to be decidable. For our domain S, we take the set of pairs $\langle e, c\rangle \in \omega \times \{c\}$, where c is the Lost Melody real of Theorem 8 and the e^{th} infinite time program outputs a presentation of a real point and no $e' < e$ is the index of a program outputting the same point. This is indeed a decidable domain: given any pair, we first check whether the second element is c (since the set $\{c\}$ is decidable), and, if so, use c to check the remaining conditions, which we can now do because c codes an ordinal α so large that every program which halts at all must halt by stage α, as seen in the proof of Theorem 8.

Given any two elements $\langle e, c\rangle$ and $\langle e', c\rangle$ of S, we need to compute their sum, product, difference, and quotient, and also to compute the relation $<$. For each of the four operations, the proof of Lemma 13 gives a program P_{e_0} which writes a presentation of the resulting real point, with e_0 being infinite time computable uniformly in e and e'. So the result of the operation is the element $\langle e_1, c\rangle$, where e_1 is the least index of a program which outputs a presentation of the same real point as e_0. We were given c itself, of course, as part of the points $\langle e, c\rangle$ and $\langle e', c\rangle$, and with c it is simple to find the least such e_1. Thus each operation is computable on the domain S. The final claim is clear, since an element of S contains an algorithm for writing out a presentation of the corresponding real point, which in turn quickly yields every digit of the binary expansion of that point. From this, the relation $<$ on S is easily computed.

For $\mathcal{R}_w^X$, one simply relativizes the entire proof (including the choice of the Lost Melody real) to the oracle X. □

Lemma 14 shows how infinite time computable model theory differs from its finite time analogue. While we have proved that the ordered field $\mathcal{R}_w$ of infinite time computable reals (i.e. the writable reals) has an infinite time computable presentation, the corresponding fact in finite time is not true, for the finite time computable reals have no finite time computable presentation.

Proposition 15. *Let $\mathcal{R}_w$ be the ordered field of infinite time computable real points, and let $\mathcal{R}_f$ be the ordered subfield of finite time computable real points. Then neither $\mathcal{R}_w$ nor $\mathcal{R}_f$ is finite time computably presentable (in domain $\mathbb{N}$), but both are computably presentable in infinite time.*

Since the rational ordered field $\mathbb{Q}$ embeds uniquely and densely into $\mathcal{R}$, it follows that every ordered subfield $\mathbb{F}$ of $\mathcal{R}$, such as $\mathcal{R}_w$ or $\mathcal{R}_f$, embeds uniquely into our presentation of $\mathcal{R}$. We show next that this unique embedding is computable.

Lemma 16. *If $\mathbb{F}$ is any computable presentation of an ordered subfield of $\mathcal{R}$, then the unique embedding of $\mathbb{F}$ into $\mathcal{R}$ is computable.*

Proof. Given any $x \in \mathbb{F}$, we may use the computable functions of $\mathbb{F}$ to systematically compute the $\mathbb{F}$-representations of the rationals $\frac{m}{2^n}$, and make comparisons of these rationals with x using the order of $\mathbb{F}$. This allows us to know the binary representation of x, and therefore also the representation of x in our presentation of $\mathcal{R}$. Thus, we have computed the unique embedding of $\mathbb{F}$ into our presentation of $\mathcal{R}$. □

Proof of Proposition 15. An infinite time computable presentation of $\mathcal{R}_w$ was shown above to exist, and $\mathcal{R}_f$ is an infinite time computable subset of the domain, since infinite time Turing machines can easily simulate finite time ones.

If $\mathbb{F}$ were a finite time computable presentation of $\mathcal{R}_f$, then given any element $x \in \mathbb{F}$, we could compute the n-th digit of the binary expansion of the real point corresponding to $\mathbb{F}$, in finite time and uniformly in x and n. If $\mathbb{F} \cong \mathcal{R}_f$, this would give a simultaneous uniform finite time computation of all finite time computable sets, which of course is impossible. If $\mathbb{F} \cong \mathcal{R}_w$, then it would give a simultaneous uniform finite time computation of all infinite time writable reals, which again is easy to diagonalize against. This completes the proof of Proposition 15. □

We note that the same diagonalization against finite time computable presentations of all finite time computable sets can be used to show that there is no infinite time writable presentation of all infinite time writable reals. Therefore we ask how it is that $\mathcal{R}_w$ is infinite time computably presentable. The answer is that while the domain of the presentation of $\mathcal{R}_w$ is a countable decidable set, it is not the image of ω under any infinite time computable function. The use of the lost-melody real c in the domain of $\mathcal{R}_w$ makes this clear, and indeed, without using c or a similar element, we could not decide in infinite time which programs output infinite time computable reals.

A concise statement of the foregoing argument is to say that there is no writable presentation of $\mathcal{R}_w$, even though there is a computable presentation. A *writable structure* is an infinite time computable structure $\mathcal{A}$ such that there exists a single writable real $r \in 2^\omega$ whose first row $r^{[0]}$ codes the entire atomic diagram of $\mathcal{A}$ and whose remaining rows name all elements of the domain of $\mathcal{A}$. That is,

$$r^{[0]} = \{\ulcorner\varphi\urcorner \ : \ \varphi \in D_a(\mathcal{A})\}$$

$$\mathrm{dom}(\mathcal{A}) = \{r^{[n]} \ : \ n \in \omega - \{0\}\}.$$

(An equivalent definition requires that $r^{[n]} \neq r^{[m]}$ whenever $0 < n < m < 1 + |\mathcal{A}|$, and $r^{[m]} = 0$ if $m > |\mathcal{A}|$.) We assume for these purposes that the language of $\mathcal{A}_A$ is also coded into ω, with $2n-1$ coding the constant symbol for the element named as $r^{[n]}$. Thus we have a computable enumeration of the elements of $\mathcal{A}$, from which it is immediate that the complete diagram of $\mathcal{A}$ is infinite time decidable. Since they allow computable searches of the entire domain, writable structures behave something like an analogue to the finite structures in the classical theory.

Let us conclude this section with a brief generalization. Let $\mathcal{R}_a$ be the structure of the real points having an accidentally writable presentation, and similarly, let $\mathcal{R}_e$ consist of those having an eventually writable presentation.

Theorem 17. $\mathcal{R}_f \prec \mathcal{R}_w \prec \mathcal{R}_e \prec \mathcal{R}_a \prec \mathcal{R}$.

Proof. The point is that each of these structures is a real closed ordered field. Before explaining this, let us first iron out a wrinkle with $\mathcal{R}_a$. In order to see even that this structure is closed under addition, it is useful to know that the set of accidentally writable reals is closed under pairing. To see this, consider the algorithm that simulates all programs on input 0, and for each accidentally writable real x observed during this master simulation, the algorithm starts another master simulation that produces all accidentally writable reals y that appear before the first appearance of x. Then, for each such y, our main algorithm writes a real coding the pair $\langle x, y\rangle$ on the scratch tape. This algorithm shows that if x and y are accidentally writable, then the pair $\langle x, y\rangle$ is also accidentally writable. Using this and the observations of Lemma 13, it now follows that $\mathcal{R}_a$ is a field.

Each of the fields is closed under square roots for its positive elements, since the digits of the square root can be systematically computed. Also, for any odd degree polynomial, one can use successive approximations (for example, by Newton's method) to find a computable root. Since the theory of real closed fields is model complete, the theorem now follows. □

One can naturally extend this theorem by oracles and have a rich lattice of relatively computable subfields of $\mathcal{R}$. Each of the extensions in the theorem is strict, by [HL00, Theorem 6.15], and it follows that each is a transcendental extension of the previous. Finally, we observe that $\mathcal{R}_w$ can have no writable transcendence basis over $\mathbb{Q}$ or $\mathcal{R}_f$, since then we would be able to produce a writable list of all writable reals, which we have observed is impossible by a simple diagonalization. Similarly, $\mathcal{R}_e$ has no eventually writable transcendence basis over $\mathcal{R}_w$ and $\mathcal{R}_a$ has no accidentally writable transcendence basis over $\mathcal{R}_e$.

3. The Infinite time computable completeness theorem

The Completeness theorem asserts that every consistent theory has a model. The finite time effective version of this asserts that any finite time decidable theory has a finite time decidable model. And in the infinite time context, at least for languages coded in $\mathbb{N}$, this proof goes through without any hitch. In fact, the infinitary context gives a slightly stronger result:

Theorem 18. *In the infinite time context, if T is a consistent theory in a computable language coded in $\mathbb{N}$ and T has a computable axiomatization, then T has a decidable computable model. In fact, such a theory has a model coded by a writable real.*

Proof. The point is that the classical Henkin construction is effective for infinite time Turing machines. Note that if T has a computable axiomatization in a language coded in $\mathbb{N}$, then it is actually decidable, since the infinite time Turing machines can search through all proofs in ω steps. We may assume that there is an infinite supply of new constant symbols, by temporarily rearranging the Gödel codes of the symbols in the original language if necessary. Enumerate the sentences in the expanded langauge as $\langle \sigma_n \mid n \in \mathbb{N}\rangle$, and build a complete consistent Henkin theory in the usual manner: at stage n, we add σ_n, if this is consistent with what we have already added to T, or else $\neg\sigma_n$, if it is not. Since T is decidable, this is computable. In addition, if σ_n has the form $\exists x\, \varphi(x)$ and we added it to the theory, then we also add $\varphi(c)$ for the first new constant symbol c that has not yet been considered. The result of this construction is a complete consistent Henkin theory

$\bar{T}$ extending T. The theory $\bar{T}$ is decidable, because for any σ, the infinite time algorithm can run the construction until σ is considered, and answer accordingly as it was added to $\bar{T}$ or not. As usual, we may use the Henkin constants to build a model of T. Specifically, let $c \equiv d$ if $\bar{T} \vdash c = d$, and define the $R([\vec{c}]) \iff \bar{T} \vdash R(\vec{c})$ and $f([\vec{c}]) = [d] \iff \bar{T} \vdash f(\vec{c}) = d$. The classical induction shows that the resulting structure $M_{\bar{T}}$ of equivalence classes satisfies $\varphi([\vec{c}])$ if and only if $\bar{T} \vdash \varphi(\vec{c})$, so this is a model of T. Finally, for any constant symbol d, one may compute the (numerically) least element of $[d]$ by simply testing each of the smaller constants c to determine whether $c \equiv d$. Thus, by replacing each equivalence class with its least member, we construct a computable presentation of $M_{\bar{T}}$. Since the underlying set of this model is contained in $\mathbb{N}$, an algorithm can write down the entire structure as a writable real. □

Many theories, including some very powerful theories, have infinite time computable axiomatizations, and so this result provides numerous interesting decidable models. For example, the theory of *true arithmetic* $\mathrm{TA} = \mathrm{Th}(\langle \mathbb{N}, +, \cdot, 0, 1, < \rangle)$ is infinite time decidable, because arithmetic truth is infinite time decidable, and so the theory $\mathrm{TA} + \{\, n < c \mid n \in \mathbb{N} \,\}$ is a computable axiomatization of the theory of the nonstandard models of true arithmetic. Similar observations establish:

Corollary 19. *There are infinite time decidable computable nonstandard models of the theories* PA*,* TA*,* ZFC*,* ZFC + *large cardinals, and so on, provided that these theories are consistent.*

The infinite time realm, therefore, lies considerably beyond the computable models of the finite time theory. What is more, as we have emphasized, the infinite time context allows for uncountable computable models and uncountable languages, which cannot be coded in $\mathbb{N}$. So Theorem 18 doesn't tell the full story. In the general context, where languages are coded in the reals, we ask whether the full infinite time analogue of the Completeness Theorem holds:

Question 20. *Does every consistent infinite time decidable theory have an infinite time decidable model? Does every such theory have an infinite time computable model?*

One of the convenient features of the classical theory, when working with a language coded in $\mathbb{N}$, is that one can enumerate the function, relation and constant symbols of the language $s_0, s_1, \ldots$ in such a way that from any symbol s_n, one can reconstruct the list $\langle s_m \mid m \leq n \rangle$ of prior symbols. This is a triviality in the context of computable languages coded in $\mathbb{N}$, because we simply enumerate the symbols in the order of their Gödel codes. Given any such code, one simply tests all the smaller natural numbers in turn to discover the list of prior codes for symbols. But in the uncountable context, a computable representation of a language may not have this feature. Let us therefore define that a computable representation $\mathcal{L}$ of a language is computably *well presented* if there is an enumeration $\langle s_\alpha \mid \alpha < \delta \rangle$ of all of the function, relation and constant symbols of the language, for some $\delta \leq \omega_1$, such that from any $\ulcorner s_\alpha \urcorner$, we can (uniformly, in infinite time) compute a code for the sequence $\langle \ulcorner s_\beta \urcorner \mid \beta \leq \alpha \rangle$ of prior symbols. In this case, we can prove the infinite time computable analogue of the Completeness Theorem.

Theorem 21. *Every consistent infinite time decidable theory in a computably well presented language has an infinite time decidable model in this language.*

We begin with a few preliminary lemmas. Let us say that a computable presentation $\mathcal{L}$ of a language admits a *computably stratified enumeration* of formulas if there is an enumeration of all $\mathcal{L}$-formulas $\langle \varphi_\alpha \mid \alpha \leq \delta \rangle$, for some $\delta \leq \omega_1$, such that from the Gödel code $\ulcorner \varphi_\alpha \urcorner$, one can (uniformly in infinite time) compute a real coding the sequence $\langle \ulcorner \varphi_\beta \urcorner \mid \beta \leq \alpha \rangle$ of Gödel codes of the prior formulas.

Lemma 21.1. *If a language $\mathcal{L}$ is computably well presented, then it admits a computably stratified enumeration of formulas.*

Proof. Suppose that a language $\mathcal{L}$ is computably well presented by the enumeration $\langle s_\alpha \mid \alpha \leq \delta \rangle$. Given a well-ordered list of function, relation and constant symbols, one can systematically produce a list of all formulas in that language, as follows. The first ω many formulas are those not using any of the symbols; the next ω many formulas are those using the first symbol only; the next ω many formulas use the second symbol and possibly the first. There is a (finite time) computable list of countably many first order formula templates, with holes for the function, constant and relation symbols, and the actual formulas are obtained by plugging codes for actual function, relation and constant symbols (of the appropriate arity) into those holes. From the presentation of the symbols, we systematically generate a list of all finite sequences of the symbols, and from these and the templates, one can generate the list of all formulas. We therefore generate the formulas in blocks of length ω, and all formulas in the α^{th} block are required to use the symbol s_α and may use earlier symbols. This defines the enumeration of the formulas $\langle \varphi_\alpha \mid \alpha \leq \gamma \rangle$.

Given any formula $\ulcorner \varphi \urcorner$, we can inspect it for the symbols s that appear in it, and from each $\ulcorner s \urcorner$, we can generate the corresponding list of prior symbols $\langle \ulcorner s_\beta \urcorner \mid \beta \leq \alpha \rangle$, where $s = s_\alpha$. By comparing the lengths of these sequences, we can tell which symbol was the last to appear in the enumeration of $\mathcal{L}$. For this maximal α, we know that φ appears in the α^{th} block of formulas. From the list of symbols $\langle \ulcorner s_\beta \urcorner \mid \beta \leq \alpha \rangle$, we can regenerate the list of formulas up to and including the α^{th} block of formulas, thereby producing the prior list of formulas $\langle \ulcorner \varphi_\xi \urcorner \mid \xi \leq \eta \rangle$, where $\varphi = \varphi_\eta$. □

A fundamental construction of first order logic is to expand a language by adding infinitely many new constant symbols. In the context of computable model theory, whether finite or infinite time, if the presentation of a language $\mathcal{L}$ already uses all the available Gödel codes, then one is forced to consider translations of the language in order to free up space in the Gödel codes to represent the expanded language. For example, even in the finite time context, if one has a model in a language with infinitely many constant symbols, and the Gödel codes of the symbols already use up all of $\mathbb{N}$, then in order to add constants to the language one seems forced to use a translation of the language. A given language can have many different computable presentations, and in general these may not be computably equivalent. For two presentations of the language, there may be no computable method of translating symbols or formulas from one representation to the other. (And this phenomenon occurs already in the finite time context.) In the infinite time context, where we represent symbols with real numbers, this phenomenon can occur even in finite languages, since the Gödel codes for a symbol may be reals that are incomparable in the infinite time Turing degrees. If we have two computable presentations $\mathcal{L}$ and $\mathcal{L}'$ of a language, and it happens that there is a computable function mapping every $\mathcal{L}'$ code for a symbol to the $\mathcal{L}$ code for the same symbol,

then we will say that $\mathcal{L}'$ is a *computable translation* of $\mathcal{L}$. In such a case, syntactic questions about $\mathcal{L}'$ can be computably reduced to syntactic questions about $\mathcal{L}$. This relation is not necessarily symmetric (because in the infinite time context, a function can be computable without its inverse being computable). If both languages are computable translations of each other, we say that the languages are computably isomorphic translations.

Lemma 21.2. *If a language $\mathcal{L}$ is computably well presented, then there is a computably isomorphic translation of it to a well presented language $\mathcal{L}_0$, preserving the order of the enumeration of symbols, and a well presented expansion $\mathcal{L}_1$ of $\mathcal{L}_0$ containing ω many new constant symbols c_s^n for every symbol s of $\mathcal{L}$, such that from $\ulcorner s\urcorner$ and n one can uniformly compute $\ulcorner c_s^n\urcorner$ and conversely.*

Proof. For each symbol s of $\mathcal{L}$, let its code in $\mathcal{L}_0$ be obtained by simply adding a 0 to the front of $\ulcorner s\urcorner$ in $\mathcal{L}$. For $\mathcal{L}_1$, the code of the constant symbol c_s^n is obtained by adding $n+1$ many 1s plus 0 to the front of $\ulcorner s\urcorner$ in $\mathcal{L}$. Thus, from $\ulcorner s\urcorner$ in $\mathcal{L}$ we can easily compute every $\ulcorner c_s^n\urcorner$ and $\ulcorner s\urcorner$ in $\mathcal{L}_1$ and vice versa. So it is clear that $\mathcal{L}_0$ is a computably isomorphic translation of the language $\mathcal{L}$. The enumeration of the symbols of $\mathcal{L}_1$ simply replaces each symbol s of $\mathcal{L}$ with the block of symbols s, c_s^0, c_s^1, and so on. From any of these symbols, we can reconstruct the prior list of symbols in $\mathcal{L}$, and from those symbols we can reconstruct the corresponding constant symbols, so as to generate the prior list of symbols in $\mathcal{L}_1$. □

Proof of Theorem 21. We carry out the proof of Theorem 18 in this more general context. Suppose that T is a computably axiomatized consistent theory in the well presented language $\mathcal{L}$. Let $\mathcal{L}'$ be the well presented language of Lemma 21.2, with infinitely many new constant symbols for each symbol of $\mathcal{L}$. Because it is well presented, this expanded language has a computably stratified enumeration $\langle \ulcorner\varphi_\alpha\urcorner \mid \alpha < \delta\rangle$ of formulas. We assume that this language is enumerated in the manner of Lemma 21.1, in blocks of length ω containing all formulas with a given symbol and earlier symbols. Because we arranged that every symbol s of $\mathcal{L}$ gives rise to an infinite list of new constant symbols c_s^n, we may arrange that from any $\ulcorner\varphi_\alpha\urcorner$, we may uniformly compute the code of a distinct new constant symbol c not appearing in any earlier φ_β.

We now recursively build the theory $\bar{T}$ in stages: at stage α, if φ_α is a sentence, then we add it to $\bar{T}$ if this remains consistent, otherwise we add $\neg\varphi_\alpha$. In addition, if φ_α is a sentence of the form $\exists x\, \psi(x)$ and we had added it to $\bar{T}$, then we also add a sentence of the form $\psi(c)$, where c is the distinct new constant symbol which has not yet appeared in any earlier formula. The usual model theoretic arguments show that $\bar{T}$ is a complete consistent Henkin theory extending T.

We argue that $\bar{T}$ is decidable. Given any $\mathcal{L}'$ formula φ_α, we may use the computable stratification to write down a code of $\langle \ulcorner\varphi_\beta\urcorner \mid \beta \leq \alpha\rangle$. From this, we may computably reconstruct $\bar{T}$ up to stage α. The question of whether to add φ_β or $\neg\varphi_\beta$ at stage β reduces to a question about whether the theory constructed up to stage β proves $\neg\varphi_\beta$ or not. But since the algorithm has a real coding the theory constructed up to stage β, it can computably enumerate all finite combinations of the formulas it is committed to adding to T, and check whether T proves that any of those finite combinations of formulas proves $\neg\varphi_\beta$. This is a decidable question, since T is decidable and we may computably translate between the languages $\mathcal{L}$ and $\mathcal{L}'$. Thus, $\bar{T}$ is computable.

Next, we build a decidable model of $\bar{T}$. Define the equivalence relation $c \equiv d \iff \bar{T} \vdash c = d$, and from each equivalence class $[c]$, select the constant $c^n_{s_\alpha}$ such that the pair $\langle \alpha, n\rangle$ is lexicographically least, where s_α is the α^{th} symbol in the original presentation of $\mathcal{L}$. The set of such least constants is decidable, because from any constant $c^n_{s_\alpha}$ we may construct the list of prior symbols, and therefore the ω-blocks of the symbols in $\mathcal{L}'$, and therefore all the corresponding formulas φ_β containing only those symbols. By reconstructing the theory $\bar{T}$ up to that point, we can tell whether $\bar{T}$ proves $c^n_{s_\alpha} = c^m_{s_\xi}$ or not, for any $\xi < \alpha$. So the set of such least representatives is decidable. We may now impose the usual structure on these representatives, to get a decidable model of $\bar{T}$. Since we have a computable isomorphism of $\mathcal{L}'$ with $\mathcal{L}$, it is no problem to translate between the two languages, and so we may use the original language presentation $\mathcal{L}$ when imposing this structure, resulting in a decidable model of T in the original language $\mathcal{L}$, as desired. □

Theorem 22. *If $V = L$, then every consistent infinite time decidable theory has an infinite time decidable model, in a computable translation of the language.*

Proof. The first step is to translate to a computably well presented language.

Lemma 22.1. *If $V = L$, then every computably presented language has a computable translation to a computably well presented language.*

Proof. Assume $V = L$ and suppose that $\mathcal{L}$ is a computably presented language. Let $S \subseteq \mathbb{R}$ be the corresponding computable set of Gödel codes for the function, relation and constant symbols of $\mathcal{L}$. Let $\langle s_\alpha \mid \alpha < \delta\rangle$ be the enumeration of the elements of S in order type $\delta \leq \omega_1$, using the canonical L-ordering of $\mathbb{R}^L$. For each $\alpha < \delta$, let γ_α be the smallest countable ordinal above α such that L_{γ_α} satisfies "ω_1 exists" and s_β exists for every $\beta \leq \alpha$. By this latter assertion, we mean that for every $\beta \leq \alpha$, the structure L_{γ_α} computes that S has at least β many elements in the L-order. Notice that because it satisfies "ω_1 exists," this structure correctly computes all infinite time computations for input reals that it has. Therefore, it correctly computes $S \cap L_{\gamma_\alpha}$, which has $\langle s_\beta \mid \beta \leq \alpha\rangle$ as an initial segment in the L order. In particular, $\langle s_\beta \mid \beta \leq \alpha\rangle \in L_{\gamma_\alpha}$. Let t_α be the L-code of the pair $\langle \alpha, \gamma_\alpha\rangle$. We will use t_α to represent the symbol coded by s_α in $\mathcal{L}$. Denote this new translation of the language by $\mathcal{L}'$.

First, we observe that the set $\{\, t_\alpha \mid \alpha \leq \delta \,\}$ is decidable. Given any real t, we can check if it is the L-code of a pair of ordinals $\langle \alpha, \gamma\rangle$, and if so, whether γ is least such that L_γ satisfies "ω_1 exists" and s_β exists for every $\beta \leq \alpha$. If so, then we accept t. Necessarily, in this case $t = t_\alpha$. These questions are all decidable, because we know how to recognize an L-code for a pair of ordinals, and given the code of an ordinal γ we can construct a code of L_γ, and then check the truth of any statement in that structure by Theorem 6.

What is more, from t_α we can construct all earlier t_β for $\beta \leq \alpha$, because with an L-code for γ we can look for the least $\gamma' \leq \gamma$ such that $L_{\gamma'}$ satisfies "ω_1 exists" and s_ξ exists for all $\xi \leq \beta$. Thus, our new language is computably well presented via $\mathcal{L}'$. Finally, $\mathcal{L}'$ is a computable translation of $\mathcal{L}$ because from t_α we can compute s_α. □

We remark that the translation from $\mathcal{L}$ to $\mathcal{L}'$, while perhaps not a computably isomorphic translation, is nevertheless relatively mild. Specifically, from s_α and

any code for a sufficiently large ordinal, one can compute t_α. In this sense, the two representations of the language are close.

We now complete the proof of Theorem 22. Assume $V = L$ and suppose that T is a consistent decidable theory in a language $\mathcal{L}$. (By testing whether certain tautologies are well formed, it follows that the language itself is computable.) By Lemma 22.1, there is a computable translation of $\mathcal{L}$ to a well presented language $\mathcal{L}'$. Let T' be the corresponding translation of T into this translated language. Note that T' remains decidable in $\mathcal{L}'$, because the question $T' \vdash \sigma'$ computably reduces to a question of the form $T \vdash \sigma$, which is decidable. By Theorem 21, the theory T' has a decidable model, as desired. □

So it is at least consistent with ZFC that the infinite time computable Completeness Theorem holds, if one allows computable translations of the language, and in this sense one may consistently hold a positive answer to Question 20. Does this settle the matter? No, for we will now turn to negative instances of the completeness theorem. The fact is that in some models of set theory, there are consistent decidable theories having no decidable model, and so the infinitary computable Completeness Theorem is actually independent of ZFC.

Theorem 23. *It is relatively consistent with* ZFC *that there is an infinite time decidable theory, in a computably presented language, having no infinite time computable or decidable model in any translation of the language (computable or not).*

This theorem relies on the following fact from descriptive set theory. For a proof, see [Jec03, Theorem 25.23].

Lemma 23.1. (Mansfield-Solovay) *If $A \subseteq \mathbb{R}$ is Σ^1_2 and $A \not\subseteq L$, then A contains a perfect subset.*

The crucial consequence for us will be:

Lemma 23.2. *If ω_1^L is countable and the* CH *fails, then there are no Σ^1_2 sets of size ω_1. Hence, under these hypotheses, there are also no decidable sets or semi-decidable sets of size ω_1.*

Proof. Every decidable or semi-decidable set $A \subseteq \mathbb{R}$ is Δ^1_2 and hence Σ^1_2. If ω_1^L is countable and $A \subseteq L$, then A is countable. If $A \not\subseteq L$, then by Lemma 23.1 it contains a perfect subset, and hence has cardinality 2^ω. Under ¬CH, this excludes the possibility that A has cardinality ω_1. □

Proof of Theorem 23. Suppose that ω_1^L is countable and the CH fails. An elementary forcing argument shows that this hypothesis is relatively consistent with ZFC. Lemma 23.2 now shows that there are no Σ^1_2 sets of size ω_1. Consider the following theory, in the language with a constant c_x for every $x \in \text{WO}$ (for simplicity, let $\ulcorner c_x \urcorner = x$), a binary relation $\equiv$ and a function symbol f. The theory T is the atomic diagram of the structure $\langle \text{WO}, \equiv \rangle$, where $\equiv$ is the relation of coding the same ordinal, together with the axiom asserting that f is a choice function on the equivalence classes. That is, T contains all the atomic facts that are true about the constants c_x for $x \in \text{WO}$, plus the assertion "$x \equiv f(x)$ and $x \equiv y \implies f(x) = f(y)$." This theory is computably axiomatizable, because $\equiv$ is a decidable relation on WO. So as a set of sentences, the axioms of T are decidable.

But actually, the theory T is fully decidable. First, we observe that it admits elimination of quantifiers. The point is that T is essentially similar to the theory of

an equivalence relation with infinitely many equivalence classes, all infinite. Note that the theory T implies $f(f(x)) = f(x)$, and $x \equiv f(y)$ is the same as $x \equiv y$. Also, $x = f(y)$ is equivalent to $x = f(x) \ \& \ x \equiv y$. By combining these reductions with the usual induction, it suffices to eliminate quantifiers from assertions of the form $\exists x\, x \equiv y \ \& \ x \not\equiv z \ \& \ x = f(x)$ and $\exists x\, x \equiv y \ \& \ x \not\equiv z \ \& \ x \neq f(x)$. But these are both equivalent to $y \not\equiv z$, since in the former case one may use $x = f(y)$, and in the latter case some x equivalent to y, other than $f(y)$. This inductive reduction provides a computable method of finding, for any given formula, a quantifier-free formula that is equivalent to it under T. The point now is that any quantifier-free sentence is a Boolean combination of assertions about the constants c_x of the form $c_x \equiv c_y$, $c_x = c_z$ and $f(c_x) = c_y$. The first two of these are computable, since they are equivalent to $x \equiv y$ and $x = z$, respectively. The assertion $f(c_x) = c_y$ is false if $x \not\equiv y$, which is computable, and otherwise it is not settled by T, since there are models of T where $f(c_x)$ is any desired c_y with $y \equiv x$. For any finite list of constants c_y, it is consistent that $f(c_x)$ is equal to any of them (at most one of them), provided $x \equiv y$, or none of them. Because of this, we can computably decide whether T proves any given quantifier-free assertion in the language of T. So T is decidable.

Finally, suppose towards contradiction that T has a computable or decidable model $M = \langle A, \equiv^M, f^M, c_x^M \rangle_{x \in \mathrm{WO}}$. In this case, both the graph of f and the relation $z = c_x^M$ are decidable, and so the set $\{\, f(c_x^M) \mid x \in \mathrm{WO} \,\}$ has complexity Σ^1_2. But this set also has cardinality ω_1, contradicting Lemma 23.2. So T can have no computable or decidable model under these set theoretic hypotheses. Since the set theoretic hypotheses are relatively consistent with ZFC, it is relatively consistent with ZFC that there is an infinite time decidable theory with no computable or decidable model. □

With Theorems 22 and 23 we have now established the following.

Theorem 24. *The infinite time computable Completeness Theorem is independent of* ZFC*.*

For this theorem, we take the infinite time computable Completeness Theorem to be the assertion: every consistent decidable theory in a computably presented language has a decidable model in a computable translation of the language.

4. The infinite time computable Lowenheim-Skolem Theorem

The classical Lowenheim-Skolem Theorem has two parts: the upward theorem asserts that every infinite model has arbitrarily large elementary extensions, in every cardinality at least as large as the original model and the language; the downward theorem asserts that every infinite model has elementary substructures of every smaller infinite cardinality at least as large as the language. Here, of course, we are interested in the infinite time computable analogues of these assertions, which concern computable or decidable models.

Question 25. *Does every infinite time decidable model have an infinite time decidable elementary extension of size continuum?*

Question 26. *Does every infinite time decidable infinite model (in a language coded in* $\mathbb{N}$*, say) have a countable infinite time decidable elementary substructure?*

These questions have many close variants, depending, for example, on whether the models are decidable or computable, and on whether the languages or models are well presented or not. One could ask in Question 25 merely for a proper elementary extension, or for an uncountable extension, rather than one of size continuum, and in Question 26, merely for a proper elementary substructure rather than a countable one (when the original model is uncountable). We regard all such variations as infinite time computable analogues of the Lowenheim-Skolem Theorem.

If the Continuum Hypothesis fails badly, then it is too much to ask for computable models of every cardinality between ω and 2^ω. To be sure, this is clearly impossible if the continuum is too large (if $2^\omega \geq \aleph_{\omega_1}$), for in this case there would be uncountably many such intermediate cardinalities but only countably many decidable models. More importantly, however, Lemma 23.1 shows that there can be no decidable sets of cardinality strictly between ω_1^L and 2^ω. Thus, the possible cardinalities of decidable sets of reals are: finite, countable, ω_1^L and 2^ω.

We do not know the full answers to either of the questions above, although we do know the answers to some of the variants. For the upward version, if a model is well presented, then we can find an infinite time decidable proper elementary extension (see Theorem 27); if $V = L$, then we can arrange this extension to be uncountable (see Theorem 28). So it is consistent that the upward Lowenheim-Skolem Theorem holds. For the downward version, if an uncountable decidable model is well presented, then we can always find a countable decidable elementary substructure (see Theorem 29); but if one broadens Question 26 to the case of computable models, rather than decidable models, then we have a strong negative answer, for there is a computable structure on $\mathbb{R}$ having no computable proper elementary substructures (see Theorem 30).

In analogy with well presented languages, let us define that an infinite time computable model $\mathcal{A} = \langle A, \cdots \rangle$ is *well presented* if the language of its elementary diagram is well presented. This means that there is an enumeration $\langle s_\alpha \mid \alpha < \delta \rangle$, for some $\delta \leq \omega_1$, including every Gödel code for a symbol in the language and every element of A, such that from s_α one can compute a code for $\langle s_\beta \mid \beta \leq \alpha \rangle$. The models produced in the computable Completeness Theorem 21, for example, have this property.

Theorem 27. *If $\mathcal{A}$ is a well presented infinite time decidable infinite model, then $\mathcal{A}$ has a proper elementary extension with an infinite time decidable presentation.*

Proof. Let T be the elementary diagram of $\mathcal{A}$, in a well presented language. Let $\mathcal{L}'$ be the language of T together with new constants, as in Lemma 21.2. Let T' be the theory T together with the assertion that these new constants are not equal to each other or to the original constants. Since T is decidable, it is easy to see that T' is decidable, since any question about whether T' proves an assertion about the new constants can be decided by replacing them with variables and the assumption that those variables are not equal. Thus, by Theorem 21, there is an infinite time decidable model of T'. Such a model provides a decidable presentation of a proper elementary extension of $\mathcal{A}$. □

Theorem 28. *If $V = L$, then every infinite time decidable infinite model $\mathcal{A}$ elementarily embeds into an infinite time decidable model of size the continuum, in a computable translation of the language.*

Proof. Assume $V = L$ and suppose that $\mathcal{A}$ is an infinite time decidable infinite model. We may assume, by taking a computably isomorphic copy of the language, that all the Gödel codes of symbols and elements in $\mathcal{A}$ begin with the digit 0. So there are continuum many additional codes, beginning with 1, that we use as the Gödel codes of new constant symbols. If T is the elementary diagram of $\mathcal{A}$, then let T' be T together with the assertion that these new constants are not equal. The theory T' is decidable, because any question about whether T' proves an assertion reduces to a question about whether T proves an assertion about some new arbitrary but unequal elements. This can be decided by replacing those new constant symbols with variable symbols plus the assertion that they are distinct. Thus, by Theorem 22, there is a decidable model $\mathcal{A}' \models T'$. The model $\mathcal{A}'$ has size continuum because of the continuum many new constants we added, and $\mathcal{A}$ embeds elementarily into $\mathcal{A}'$ because $\mathcal{A}'$ satisfies the elementary diagram of $\mathcal{A}$. □

We note that the graph of the elementary embedding of $\mathcal{A}$ into $\mathcal{A}'$ is infinite time decidable, because from the code of a symbol in the expanded language, one can compute the code of the corresponding symbol in the original language. There seems little reason to expect in general that this embedding should be a computable function, and it cannot be if the original presentation was not well presented.

Let us turn now to the infinite time computable analogues of the downward Lowenheim-Skolem Theorem.

Theorem 29. *If $\mathcal{A}$ is an uncountable well presented infinite time decidable model in a language coded by a writable real, then there is an infinite time decidable, countable elementary substructure $\mathcal{B} \prec \mathcal{A}$.*

Proof. The idea is to effectively verify the Tarski-Vaught criterion on the shortest initial elementary cut of the well presented enumeration of $\mathcal{A}$. So, suppose that $\langle a_\alpha \mid \alpha < \omega_1 \rangle$ is the well presented enumeration of the underlying set of $\mathcal{A}$. By classical methods, there is a closed unbounded set of countable initial segments of this enumeration that form elementary substructures of $\mathcal{A}$. Let β be least such that $B = \{ a_\alpha \mid \alpha < \beta \}$ forms an elementary substructure $\mathcal{B} \prec \mathcal{A}$. Thus, β is least such that the set $\{ a_\alpha \mid \alpha < \beta \}$ satisfies the Tarski-Vaught criterion in $\mathcal{A}$. We will argue that B is infinite time decidable as a set. Given any a_ξ, we can generate the sequence $\langle a_\alpha \mid \alpha < \xi \rangle$ and for each $\xi' \leq \xi$ we can check whether $\{ a_\alpha \mid \alpha < \xi' \}$ satisfies the Tarski-Vaught criterion in $\mathcal{A}$. To check this, we use the writable real coding the language to generate a list of all formulas φ in the language. For every such formula φ and every finite sequence $a_{\alpha_0}, \ldots, a_{\alpha_n}$ with each $\alpha_i < \xi'$, we use the decidability of $\mathcal{A}$ to inquire whether $\exists x\, \varphi(x, a_{\alpha_0}, \ldots, a_{\alpha_n})$ is true in $\mathcal{A}$. If so, then we check that there is some $\alpha < \xi'$ with $\varphi(a_\alpha, a_{\alpha_0}, \ldots, a_{\alpha_n})$ true in $\mathcal{A}$. These checks will all be satisfied if and only if $\{ a_\alpha \mid \alpha < \xi' \}$ satisfies the Tarski-Vaught criterion. Consequently, if such a ξ' exists with $\xi' \leq \xi$, then by the minimality of β, it must be that $\beta \leq \xi'$, and so a_ξ is not in B. If no such ξ' exists up to ξ, then $\xi < \beta$ and so $a_\xi \in B$. Therefore, as a set, B is decidable. The corresponding model $\mathcal{B}$ is therefore a decidable model, and a countable elementary substructure of $\mathcal{A}$, as desired. □

Finally, we have a strong violation to the infinite time computable downward Lowenheim-Skolem Theorem, when it comes to computable models. For infinite

time Turing machines, a *computation snapshot* is a real coding the complete description of a machine configuration, namely, the program that the machine is running, the head position, the state and the contents of the cells.

Theorem 30. *There is an infinite time computable structure with underlying set $\mathbb{R}$ having no infinite time computable proper elementary substructure.*

Proof. Define the relation $U_p(x, y)$ if y codes the computation sequence of program p on input x showing it to have been accepted. That is, y codes a well-ordered sequence of computation snapshots $\langle y_\alpha \mid \alpha \leq \beta\rangle$, such that (i) the first snapshot y_0 is the starting configuration of the computation of program p on input x; (ii) successor snapshots $y_{\alpha+1}$ are updated correctly from the prior snapshot y_α and the operation of p; (iii) limit snapshots y_ξ correctly show the head on the left-most cell in the *limit* state, with the tape updated correctly from the prior tape values in $\langle y_\alpha \mid \alpha < \xi\rangle$; and lastly, (iv) the final snapshot y_β shows that the computation halted and accepted the input. This is a computable property of $\langle p, x, y\rangle$, since one can computably verify that y codes such a well ordered sequence of snapshots by counting through the underlying order of y and systematically checking each of the requirements. So the structure $\mathcal{R} = \langle \mathbb{R}, U_p\rangle_{p\in\mathbb{N}}$ is a computable structure. (One could reduce this to a finite language with a trinary predicate $U(p, x, y)$, by regarding programs as reals and ensuring that the programs are necessarily in any elementary substructure.)

Suppose that there is a computable proper elementary substructure $\mathcal{A} \prec \mathcal{R}$. Let p_0 be a program deciding the underlying set A of $\mathcal{A}$. Since every real $a \in A$ is accepted by p_0, there will be a real y in $\mathbb{R}$ coding the computation sequence and witnessing $U_{p_0}(a, y)$. Thus, $\mathcal{A} \models \forall a\, \exists y\, U_{p_0}(a, y)$. By elementarity $\mathcal{A} \prec \mathcal{R}$, we conclude that $\mathcal{R}$ also satisfies this assertion. So every real is accepted by p_0. Thus, $A = \mathbb{R}$ and the substructure is not a proper substructure after all. □

Since this model is only infinite time computable and not infinite time decidable (the halting problem 0^∇ is expressible in the Σ_1 diagram), the following question remains open:

Question 31. *Is there an infinite time decidable model with underlying set $\mathbb{R}$ having no proper infinite time computable elementary substructure?*

Such a model would be a very strong counterexample to the infinite time computable downward Lowenheim-Skolem Theorem.

5. Computable quotient presentations

Recall from Definition 1 that a structure has an infinite time *computable presentation* if it is isomorphic to an infinite time computable structure. Weakening this concept slightly, let us define that a structure $\mathcal{A}$ has an infinite time computable *quotient presentation* if there is an infinite time computable structure $\mathcal{B} = \langle B, \ldots\rangle$ and an infinite time computable equivalence relation $\equiv$ on B such that $\mathcal{A}$ is isomorphic to the quotient structure $\mathcal{B}/\!\equiv$. In particular, $\equiv$ should be a *congruence relation* on $\mathcal{B}$, meaning that the functions and relations of $\mathcal{B}$ are well defined on the $\equiv$-equivalence classes $[b]_\equiv$ for $b \in B$, while the quotient structure $\mathcal{B}/\!\equiv$ consists precisely of these equivalence classes with the induced functions and relations.

Every computable structure, of course, has a computable quotient presentation, using the equivalence relation of identity. Other more elaborate and interesting

quotient presentations involve nontrivial equivalence relations. The difference between the two kinds of presentation has to do with the two possibilities in first order logic of treating $=$ as a logical symbol, insisting that it be interpreted as identity in a model, or treating it axiomatically, so that it can be interpreted merely as an equivalence relation. The natural question here, of course, is whether the two notions coincide.

Question 32. *Does every structure with an infinite time computable quotient presentation have an infinite time computable presentation?*

This is certainly true in the context of finite time computability, because one can build a computable presentation by using the least element of each equivalence class. More generally, for the same reason, it is true in the infinite time context for structures having a quotient presentation whose underlying set is contained in the natural numbers. Specifically, if $\mathcal{A} = \langle A, \ldots, \equiv \rangle$ is computable and $A \subseteq \mathbb{N}$, where $\equiv$ is a congruence on $\mathcal{A}$, then $\mathcal{A}/\equiv$ has a computable presentation. This is because the function s, mapping every $n \in A$ to the least element $s(n)$ in the equivalence class of n, is computable. To compute $s(n)$, one may simply try out all the smaller values in turn to discover the least representative. It follows that the set $B = \{\, s(n) \mid n \in A \,\}$ is a computable choice set for the collection of equivalence classes. For any relation symbol R in the language of $\mathcal{A}$, we may now naturally define $R^{\mathcal{B}}(\vec{n}) \iff R^{\mathcal{A}}(\vec{n})$; and for any function symbol f we define $f^{\mathcal{B}}(\vec{n}) = s(f^{\mathcal{A}}(\vec{n}))$. These are clearly computable functions and relations, and since $\equiv$ is a congruence, it follows that $\mathcal{A}/\equiv$ is isomorphic to $\mathcal{B}$, as desired. This argument shows more generally that if a structure has a computable quotient presentation $\langle A, \ldots, \equiv \rangle$, and there is a computable function s mapping every element to a representative for its equivalence class, then the quotient structure $\mathcal{A}/\equiv$ has a computable presentation. (Note: the range of such a computable choice function will be decidable, because it is precisely the collection of x in the original structure for which $s(x) = x$.) Such a function s is like a computable choice function on the equivalence classes.

In the general infinite time context, of course, one does not expect necessarily to be able to effectively compute representatives from each equivalence class. In fact, we will show that the answer to Question 32 is independent of ZFC. In order to illustrate the ideas, let us begin with the simple example of the uncountable well order $\langle \omega_1, < \rangle$.

Theorem 33.

(1) *The uncountable well-ordered structure $\langle \omega_1, < \rangle$ has an infinite time computable quotient presentation.*
(2) *It is relatively consistent with* ZFC *that $\langle \omega_1, < \rangle$ has no infinite time computable presentation.*

Proof. For the first claim, observe that the structure $\langle \mathrm{WO}, <, \equiv \rangle$ is an infinite time computable quotient presentation of $\langle \omega_1, < \rangle$. For any $x \in \mathrm{WO}$, the equivalence class $[x]_{\equiv}$ is exactly the set of reals coding the same ordinal as x, and so $\langle \mathrm{WO}, < \rangle / \equiv$ is isomorphic to $\langle \omega_1, < \rangle$, as desired.

For the second claim, observe that by forcing, one may easily collapse ω_1^L and add sufficient Cohen generic reals, so that in the forcing extension $V[G]$ we have that ω_1^L is countable and the CH fails. By Lemma 23.2, therefore, the model $V[G]$ has no computable structures of size ω_1. In particular, in $V[G]$ the structure $\langle \omega_1, < \rangle$ has no computable presentation, as desired. □

Thus, it is consistent that the answer to Question 32 is negative. We turn now to the possibility of a positive answer. Let us begin with a positive answer for the specific structure $\langle \omega_1, < \rangle$.

Theorem 34. *If $\omega_1 = \omega_1^L$ (a consequence of $V = L$), then the structure $\langle \omega_1, < \rangle$ has an infinite time computable presentation.*

Proof. We already observed after Theorem 12 that $\langle \omega_1^L, < \rangle$ has a computable presentation using the L-codes for ordinals. □

It seems likely that one doesn't really need the failure of CH in the proof of Theorem 33, and we suspect that the particular structure $\langle \omega_1, < \rangle$ has a computable presentation if and only if $\omega_1 = \omega_1^L$. That is, we suspect that the converse of Theorem 34 also holds.

Corollary 35. *The question of whether the structure $\langle \omega_1, < \rangle$ has an infinite time computable presentation is independent of* ZFC.

Proof. On the one hand, by Theorem 33 it is relatively consistent that $\langle \omega_1, < \rangle$ has no computable presentation. On the other hand, if $V = L$ or merely $\omega_1^L = \omega_1$, then $\langle \omega_1, < \rangle$ has a computable presentation. □

Rather than studying just one structure, however, let us now turn to the possibility of a full positive solution to Question 32. Under $V = L$, one has a full affirmative answer.

Theorem 36. *If $V = L$, then every structure with an infinite time computable quotient presentation has an infinite time computable presentation.*

Proof. Assume $V = L$ and suppose that $\mathcal{A} = \langle A, \ldots, \equiv \rangle$ is a computable structure, where $\equiv$ is a congruence with respect to the rest of the structure. We would like to show that $\mathcal{A}/\equiv$ has a computable presentation. Our argument will be guided by the idea of building a computable presentation of $\mathcal{A}/\equiv$ by selecting the L-least representatives of each equivalence class. We will not, however, be able to do exactly this, because we may not be able to recognize that a given real is the L-least representative of its equivalence class. Instead, we will attach an escort y to every such L-least representative x of an equivalence class $[x]$, where y codes an ordinal sufficiently large to allow us computably to verify that x is the L-least representative of its equivalence class. We will then build the computable presentation out of these escorted pairs $\langle x, y \rangle$.

First, for simplicity, consider the case that $\mathcal{A}$ is a relational structure. Let B be the set of pairs $\langle x, y \rangle$ such that y is an L-code for the least ordinal α such that x is an element of L_α and L_α satisfies that x is in A, that "ω_1 exists" and that x is the L-least real that is equivalent to x. The assertions about membership in A or equivalence can be expressed in L_α using the programs that compute these relations. Note that because $L_\alpha \models$ "ω_1 exists," all the computations for reals in L_α either halt or repeat before α, and so L_α has access to the full, correct computations for the reals in L_α.

We claim that B is decidable. First, the set of L-codes is decidable. Next, given that y is the L-code of an ordinal α, we can by Theorem 7 compute a code for the whole structure L_α, and so questions of satisfaction in this structure will be decidable. Next, we can check that x is an element of L_α, and that L_α satisfies all

those other properties, as desired. Checking that α is least with those properties amounts to checking that L_α thinks there is no β having an L-code that works.

Next, observe that if $\langle x, y\rangle \in B$, then x really is the L-least representative of $[x]$ in $\mathcal{A}$. The reason is that if $z \equiv x$ and z precedes x in the L order, then z would be in L_α also, where y codes α, and so L_α would know that z precedes x. And it is correct about whether $z \equiv x$, since it has the computation checking this. The point is that L_α can see x and all its L predecessors, and it knows whether they are equivalent or not. So L_α will be correct about whether x is the L-least representative of $[x]$.

Finally, we put a structure on B as follows. For a relation symbol R, let $R^{\mathcal{B}}(\langle x_0, y_0\rangle, \ldots, \langle x_n, y_n\rangle)$ hold if and only if $R^{\mathcal{A}}(x_0, \ldots, x_n)$, which is computable. For each $a \in A$, there is an L-least representative x in $[a]$, and a least ordinal α large enough so that x is in L_α and L_α satisfies all those tests. If y is the L-code of α, then $\langle x, y\rangle$ will be in B. By mapping $[a]$ to $\langle x, y\rangle$, it is clear that $\mathcal{A}/{\equiv}$ is isomorphic to $\mathcal{B}$, providing a computable presentation.

When the language has function symbols, we define $f^{\mathcal{B}}(\langle x_0, y_0\rangle, \ldots, \langle x_n, y_n\rangle) = \langle x, y\rangle$, where x is the L-least member of $f^{\mathcal{A}}(x_0, \ldots, x_n)$ and y is the L-code for which $\langle x, y\rangle \in B$. The point now is that since $f^{\mathcal{A}}(x_0, \ldots, x_n)$ is the result of a computation in L_α, where α is the largest of the ordinals arising from $y_0, \ldots, y_n$, with the structure L_α, we will be able to find the L-least member x of the corresponding equivalence class and the L-code y putting $\langle x, y\rangle$ into B. Thus, we will be able to compute this information from $\langle x_0, y_0\rangle, \ldots, \langle x_n, y_n\rangle$, and so $f^{\mathcal{B}}$ is a computable function. Once again $\mathcal{A}/{\equiv}$ is isomorphic to $\mathcal{B}$, as desired. □

The argument does not fully use the hypothesis that $V = L$, but rather only that $A \subseteq L$, since in this case we might as well live inside L. In particular, any structure that has a computable quotient presentation using only writable reals or even accidentally writable reals, has a computable presentation.

Corollary 37. *The answer to Question 32 is independent of* ZFC.

Proof. By Theorem 33, it is relatively consistent that there is a structure with a computable quotient presentation, but no computable presentation. On the other hand, by Theorem 36, it is also relatively consistent that every structure with a computable quotient presentation has a computable presentation. □

Another way to express what the argument shows is the following. Let us say that a function $f : \mathbb{R} \to \mathbb{R}$ is *semi-computable* if its graph is semi-decidable.

Theorem 38. *If $V = L$ and $\equiv$ is an infinite time computable equivalence relation on a decidable set, then there is a semi-computable function f such that $x \equiv y$ if and only if $f(x) = f(y)$. Succinctly, every computable equivalence relation on a decidable set reduces to equality via a semi-computable function.*

Proof. Suppose $\equiv$ is an infinite time decidable equivalence relation on $\mathbb{R}^L$. Let $f(u) = \langle x, y\rangle$ where x is the L-least member of the equivalence class $[u]_\equiv$ and y is the L-code of the least α such that $x \in L_\alpha \models$ "ω_1 exists." The relation $f(u) = \langle x, y\rangle$ is decidable, since given u and $\langle x, y\rangle$, we can computably verify that $u \equiv x$ and that y is the L-code of an ordinal α; if so, we can compute a code for L_α, and from this code we can check whether α is least such that $x \in L_\alpha \models$ "ω_1 exists" and x is the L-least member of its equivalence class. The structure L_α is correct about this because it has all the earlier reals in the L-order and it has the full computations

determining whether they are equivalent to x. So f is semi-computable. Finally, notice that $u \equiv v$ if and only if $f(u) = f(v)$, since the value of f depended only on the equivalence classes $[u] = [v]$. □

This observation opens up a number of natural questions for further analysis. One naturally wants to consider computable reductions, for example, rather than semi-computable reductions. What is the nature of the resulting reducibility hierarchy? To what extent does it share the features of the hierarchy of Borel equivalence relations under Borel reducibility? For starters, can one show that there is no computable reduction of the relation E_0 (eventual equality of two binary strings) to equality?

On a different topic, Theorem 36 will allow us to show that a positive answer to the following question is consistent with ZFC.

Question 39. *Does every infinite time decidable structure have an infinite time computable presentation?*

While this question remains open, we offer two partial solutions. First, we show in Theorem 40 that when the language is particularly simple, the answer is affirmative. Second, we show in Theorem 41 that a fully general affirmative answer, for all languages, is consistent with ZFC. We don't know whether a negative answer is consistent with ZFC.

Theorem 40. *In a purely relational language, or in a language with only relation symbols plus one unary function symbol, every infinite time decidable model has an infinite time computable presentation.*

Proof. In a purely relational language, every decidable structure is already computable. So let us suppose that $\mathcal{A}$ is an infinite time decidable structure in a language with relation symbols plus one unary function symbol f. We assume that the language is computably presented, so that $\{\ulcorner f \urcorner\}$ is decidable. For each $a \in \mathcal{A}$, let a^* be the real coding the list $\langle a, \ulcorner f \urcorner, f(a), f^2(a), f^3(a), \ldots\rangle$. Let $\mathcal{A}^*$ be the set of all such a^*. This is an infinite time decidable set, because if we are given a real x coding $\langle x_0, x_1, x_2, \ldots\rangle$, we can check whether $x_0 \in \mathcal{A}$ using the fact that the underlying set of $\mathcal{A}$ is decidable; we can check whether $x_1 = \ulcorner f \urcorner$ using the decision algorithm for the language, and after this, we can check whether $x_2 = f(x_0)$, $x_3 = f(x_2)$ and so on, using the decidability of $\mathcal{A}$. So we can check whether $x = a^*$ for some a. Next, we put a structure on $\mathcal{A}^*$. For each relation symbol U of $\mathcal{A}$, define U on $\mathcal{A}^*$ by $U(a_1^*, \ldots, a_n^*)$ if and only if $U(a_1, \ldots, a_n)$. This is computable because a is computable from a^*. Next, define $f^{\mathcal{A}^*}(a^*) = (f(a))^* = \langle f(a), \ulcorner f \urcorner, f^2(a), f^3(a), f^4(a), \ldots\rangle$. The point is that this is computable from a^*, since a^* lists all this information directly. So the structure $\mathcal{A}^*$ is computable (and decidable). Since $a \mapsto a^*$ is clearly an isomorphism, this proves the theorem. □

If the language involves countably many unary function symbols and there is a writable real listing the Gödel codes of these function symbols, then a similar construction, using $a^* = \oplus\{\, \tau(a) \mid \tau \text{ is a term} \,\}$, would provide a computable presentation. This idea, however, does not seem to work with binary function symbols.

Theorem 41. *It is relatively consistent with* ZFC *that all infinite time decidable structures are infinite time computably presentable. Thus, it is consistent with* ZFC *that the answer to Question 39 is yes.*

Proof. Suppose that $\mathcal{A}$ is an infinite time decidable structure. Augment the language by adding a constant symbol for every element of $\mathcal{A}$, and let $\mathcal{A}^*$ be the set of all terms in this expanded language. The function symbols have their obvious interpretations and are computable; the relations have their natural interpretations and are decidable (since $\mathcal{A}$ is decidable). Define $t_1 \equiv t_2$ if $\mathcal{A} \models t_1 = t_2$. This is a computable equivalence relation, because $\mathcal{A}$ is decidable. Since $\mathcal{A}^*/\equiv$ is isomorphic to $\mathcal{A}$, we have provided an infinite time computable quotient presentation for $\mathcal{A}$. By Theorem 36, it is relatively consistent with ZFC that all such structures have a computable presentation. □

We note that in Theorem 41, the computable presentation may involve a computable translation of the language.

6. The infinite time analogue of Schröder-Cantor-Bernstein-Myhill

In this section, we prove the infinite time computable analogues of the Schröder-Cantor-Bernstein theorem and Myhill's theorem. With the appropriate hypotheses, as in Theorems 46 and 47, the proofs go through with a classical argument. But let us first discuss the need for careful hypotheses. The usual proofs of Myhill's theorem and the Cantor-Schröder-Bernstein theorem involve iteratively applying the functions in a zig-zag pattern between the two sets. And one of the useful properties of computable injective functions in the classical finite time context is that their corresponding inverse functions are also automatically computable: to compute $f^{-1}(b)$, one simply searches the domain for an a such that $f(a) = b$. Unfortunately, this method does not work in the infinite time context, where we generally have no ability to effectively enumerate the domain, and indeed, there are infinite time *one-way* computable functions f, meaning that f is computable but f^{-1} is not computable. An easy example of such a function is provided by the Lost Melody Theorem 8, where we have a real c such that $\{c\}$ is decidable, but c is not writable. It follows that the function $c \mapsto 1$ on the singleton domain $\{c\}$ is computable, but its inverse is not. Building on this, we can provide a decidable counterexample to a direct infinitary computable analogue of Myhill's theorem.

Theorem 42. *In the infinite time context, there are decidable sets A and B with computable total injections $f : \mathbb{R} \to \mathbb{R}$ and $g : \mathbb{R} \to \mathbb{R}$ such that $x \in B \iff f(x) \in A$ and $x \in A \iff g(x) \in B$, but there is no computable bijection $h : A \to B$.*

Proof. Let $A = \mathbb{N}$ and $B = \mathbb{N} \cup \{c\}$, where c is the real of the Lost Melody Theorem. Define $f(c) = 0$, $f(n) = n + 1$ for $n \in \mathbb{N}$ and otherwise $f(x) = x$. Clearly, f is a computable total injection and $x \in B \iff f(x) \in A$. To help define g, for any real x (infinite binary sequence), let x^* be the real obtained by omitting the first digit, and let $x^{*(n)}$ be the real obtained by omitting the first n digits. Now let $g(c) = c^*$ and more generally $g(c^{*(n)}) = c^{*(n+1)}$, and otherwise $g(x) = x$. This function g is clearly total and injective, and it is computable because given any x, we can by adding various finite binary strings to the front of x determine whether $x = c^{*(n)}$ for some n and thereby compute $g(x)$. Since c is not periodic, we have $c \notin \operatorname{ran}(g)$ and $x \in A \iff g(x) \in B$. Finally, there can be no computable onto map from A to B, since c is not the output of any computable function with natural number input. □

In this example, the function $f \restriction B$ is actually a computable bijection in the converse direction, from B to A, but this doesn't contradict the theorem because f^{-1} is not computable from A to B, since it maps 0 to c. What we really want in the infinitary context is not merely a computable bijection from A to B, but rather a computable bijection whose inverse is also computable, so that the relation is symmetric. The next example shows that we cannot achieve this even when we have computable bijections in both directions.

Theorem 43. *In the infinite time context, there are decidable sets A and B with computable bijections $f : A \longrightarrow B$ and $g : B \longrightarrow A$, for which there is no computable bijection $h : A \longrightarrow B$ whose inverse h^{-1} is also computable.*

To construct A and B, we first generalize the Lost Melody Theorem by recursively building a sequence of reals which can each be recognized, but not written, by an infinite time Turing machine using the preceding reals of the sequence.

Lemma 43.1. *There exists a sequence $\langle d_k \mid k \in \omega \rangle$ of reals such that*

1. *for each k, the real d_k is not writable from $\langle d_i \mid i < k \rangle$ and*
2. *there is an infinite time program which, for any z and any k, can decide on input $\langle d_0, d_1, \ldots, d_{k-1}, z \rangle$ whether $z = d_k$.*

Proof. The repeat-point of a computation is the least ordinal stage by which the computation either halts or enters a repeating loop from which it never emerges. For each $k \geq 0$, let δ_k be the supremum of the repeat-points of all computations of the form $\varphi_p(\langle d_i \mid i < k \rangle)$. Note that δ_k is countable in L. Let β_k be the smallest ordinal greater than δ_k such that $L_{\beta_k+1} \models$ 'β_k is countable'. Finally, let d_k be the L-least real coding β_k. The real d_k is not writable on input $\langle d_i \mid i < k \rangle$, for if it were, then we could solve the halting problem relative to $\langle d_i \mid i < k \rangle$ by writing d_k and using it to check whether any given program halts within β_k steps on input $\langle d_i \mid i < k \rangle$. Next, on input $\langle d_0, d_1, \ldots, d_{k-1}, z \rangle$, let us explain how to determine whether $z = d_k$. We first check whether z codes an ordinal α, and if so, we simulate every computation $\varphi_p(\langle d_i \mid i < k \rangle)$ for α many steps. By inspecting these computations, we can verify that they all halt or repeat by stage α, and thereby verify that $\alpha \geq \delta_k$. By Theorem 7, we can now write down a real coding $L_{\alpha+1}$ and verify that z is the L-least code for α in $L_{\alpha+1}$. If all these tests are passed, then $z = d_k$. □

Proof of Theorem 43. We use the sequence $\langle d_k \mid k \in \omega \rangle$ to construct a bi-infinite sequence $\langle c_k \mid k \in \mathbb{Z} \rangle$ as follows: for $k > 0$ let c_k be a real coding $\langle d_i \mid i < k \rangle$ in the usual manner, and for $k \leq 0$, let $c_k = k$. Let $A = \{c_{2k} \mid k \in \mathbb{Z}\}$ and $B = \{c_{2k+1} \mid k \in \mathbb{Z}\}$, and define bijections $f : A \to B$ by $f : c_{2k} \mapsto c_{2k-1}$ and $g : B \to A$ by $f : c_{2k+1} \mapsto c_{2k}$. It follows immediately from the definition of c_k that f and g are computable.

We next show that A is decidable. Given a real z, we first verify that either z is an even integer less than or equal to zero, in which case we accept it immediately, or else it codes a sequence $\langle z_0, \ldots z_{n-1} \rangle$ of even length, in which case we use the lemma iteratively to verify that $z_i = d_i$ for each $i < n$. Since the real z is an element of A if and only if it passes this test, A is decidable. Similarly, B is decidable.

We conclude by showing that if $h : A \longrightarrow B$ is a bijection, then h and h^{-1} cannot both be computable. From clause (1) of the lemma and the definition of c_n, it follows that for positive n, c_n cannot be written by any machine on input c_k

if $k < n$. Thus, if h is computable, then $h(c_2)$ must equal c_k for some $k < 2$. But then $h^{-1}(c_k) = c_2$ so h^{-1} is not computable. □

Corollary 44. *In the infinite time context, there are decidable sets A and B and a computable permutation $\pi : \mathbb{R} \to \mathbb{R}$ such that π " $A = B$ and π " $B = A$, but there is no computable bijection $h : A \to B$ for which h^{-1} is also computable.*

Proof. Let A and B be as in the proof of Theorem 43. Since A and B are disjoint, the function $\pi = f \cup g \cup \mathrm{id}$, where we use the identity function outside $A \cup B$, is a permutation of $\mathbb{R}$. Since f and g are computable and A and B are decidable, it follows that π is computable. Since π " $A = f$ " $A = B$ and π " $B = g$ " $B = A$, the proof is completed by mentioning that Theorem 43 shows that there is no computable bijection from A to B whose inverse is also computable. □

If one assumes merely that the inverses of the injections are computable, then this is insufficient to get a computable bijection:

Theorem 45. *In the infinite time context, there are semi-decidable sets A and B with computable injections $f : A \to B$ and $g : B \to A$ whose inverses are also computable, such that there is no computable bijection $h : A \to B$.*

Proof. In fact, there will be no computable surjection from A to B. Let $A = \mathbb{N}$ be the set of all natural numbers, and let $B = 0^{\triangledown} = \{\, p \mid \varphi_p(0) \downarrow \}$ be the infinite time halting problem. Define an injective function $f : A \to B$ by setting $f(n)$ to be the n^{th} program on a decidable list of obviously halting programs (such as the program with n states and all transitions leading immediately to the *halt* state). The function f is clearly computable, and by design its inverse is also computable and $\mathrm{ran}(f)$ is decidable. Conversely, construing programs as natural numbers, the inclusion map $g : B \to A$ is a computable injection whose inverse is also computable, since $\mathrm{dom}(g^{-1}) = 0^{\triangledown}$ is semi-decidable. So we have defined the required computable injections. Suppose now that $h : A \to B$ is a computable surjection of $\mathbb{N}$ to $0^{\triangledown}$. In this case, an infinite time computable function could systematically compute all the values $h(0)$, $h(1)$, and so on, and thereby write $0^{\triangledown}$ on the tape. This contradicts the fact that $0^{\triangledown}$ is not a writable real. So there can be no such computable bijection from A to B. □

In the classical finite time context, of course, there is a computable bijection between $\mathbb{N}$ and $0'$ (or any infinite c.e. set), mapping each n to the n^{th} element appearing in the canonical enumeration of it. This idea does not work in the infinitary context, however, because the infinitary halting problem $0^{\triangledown}$ is not computably enumerated in order type ω, but rather in the order type λ of the clockable ordinals. And λ is not a writable ordinal, so there is no way to effectively produce a real coding it.

Finally, with the right hypotheses, we prove the positive results, starting with the effective content of the Cantor-Schröder-Bernstein theorem.

Theorem 46. *In the infinite time context, suppose that A and B are semi-decidable sets, with computable injections $f : A \to B$ and $g : B \to A$, whose inverses are computable and whose ranges are decidable. Then there is a computable bijection $h : A \to B$ whose inverse is computable.*

Proof. Let A_0 be the set of a such that there is some finite zig-zag pre-image $(g^{-1}f^{-1})^k g^{-1}(a) \notin \mathrm{ran}(f)$ for $k \in \mathbb{N}$. Our hypotheses ensure that this set is infinite

time decidable, since we can systematically check all the corresponding pre-images to see that when and if they stop it was because they landed outside $\mathrm{ran}(f)$ in B. The usual proof of the Cantor-Schröder-Bernstein theorem now shows that the function $h = (g^{-1} \restriction A_0) \cup (f \restriction A \setminus A_0)$ is a bijection between A and B. Note that h is computable because g^{-1} and f are each computable, A_0 is decidable, and A is semi-decidable. To see that h^{-1} is computable, let $B_0 = g^{-1}A_0$ and observe that $h^{-1} = (g \restriction B_0) \cup (f^{-1} \restriction B \setminus B_0)$. Since these components are each computable, h^{-1} is computable and the proof is complete. □

We may drop the assumption that A and B are semi-decidable if we make the move to total functions, as in the classical Myhill's theorem. Define that a set of reals A is *reducible* to another set B by the function $f : \mathbb{R} \to \mathbb{R}$ if $x \in A \iff f(x) \in B$.

Theorem 47. *In the infinite time context, suppose that A and B are reducible to each other by computable one-to-one total functions f and g, whose inverses are computable and whose ranges are decidable. Then there is a computable permutation $\pi : \mathbb{R} \to \mathbb{R}$ with π^{-1} also computable and π " $A = B$.*

Proof. As in Theorem 46, let A_0 be the set of a such that some finite zig-zag pre-image $(g^{-1}f^{-1})^k g^{-1}(a) \notin \mathrm{ran}(f)$ for $k \in \mathbb{N}$, and again this is decidable. Let $\pi = (g^{-1} \restriction A_0) \cup (f \restriction \mathbb{R} \setminus A_0)$. The usual Cantor-Schröder-Bernstein argument shows that this is a permutation of $\mathbb{R}$. As above, both π and π^{-1} are computable. Finally, we have both $x \in A \iff \pi(x) \in B$ and $x \in B \iff \pi(x) \in A$, since $\pi(x)$ is either $f(x)$ or $g^{-1}(x)$, both of which have the desired properties. It follows that π " $A = B$ and the proof is complete. □

7. Some infinite time computable transitive models of set theory

Because the power of the machines are intimately connected with well-orders and countable ordinals, it is not surprising that there are many interesting models of a set theoretic nature. We have already seen that the hereditarily countable sets have an infinite time computable quotient presentation $\langle \mathrm{HC}, \in, \equiv \rangle$. In addition, we have provided infinite time computable presentations of the model $\langle L_{\omega_1^L}, \in \rangle$ and of $\langle L_\alpha, \in \rangle$, given a real coding α. In this section we will show, however, that depending on the set theoretic background, one can transcend these, by actually producing infinite time decidable presentations of *transitive* models of ZFC, or even ZFC plus large cardinals. Each of these presentations, however, will involve a somewhat strange manner of coding information into the individual elements of the model or into the language, even while the model and language technically remains computable. We begin by proving that the task is impossible without such subterfuge.

Theorem 48. *There is no infinite time computable presentation of a transitive model of* ZFC *with underlying set $\mathbb{N}$ and Gödel codes of the language entirely in $\mathbb{N}$.*

Proof. The operation of an infinite time Turing machine is absolute to any transitive model of ZFC containing the input. Thus, all transitive models of ZFC agree on the elements of the halting problem $0^\triangledown$. If $\mathcal{M} = \langle \mathbb{N}, E \rangle$ is a computable presentation of such a model, then there is some natural number k representing $0^\triangledown$ in $\mathcal{M}$. Assuming that $\ulcorner \in \urcorner$ is writable, then we can computably determine for each natural number

p the element k_p representing it in $\mathcal{M}$. In this case, we could compute $0^\nabla = \{\, p \mid k_p \; E \; k \,\}$, contradicting the fact that 0^∇ is not computable. □

Of course, this argument uses much less than ZFC. It shows that there can be no computable presentation, using underlying set $\mathbb{N}$ and writable presentation of the language, of a transitive model computing 0^∇ correctly. For example, it would be enough if the model satisfied "ω_1 exists," or even less, that every infinite time Turing computation either halted or reached its repeat point.

Despite Theorem 48, however, computable models and languages are not in general limited to the domain $\mathbb{N}$, and in this general setting we can actually find computable presentations of transitive well founded models of ZFC.

Theorem 49. *If there is a transitive model of* ZFC*, then the smallest transitive model of* ZFC *has an infinite time decidable computable presentation.*

Proof. If there is a transitive model of ZFC, then there is one satisfying $V = L$. A Löwenheim-Skolem argument, followed by the Mostowski collapse, shows that there must be a countable such model, and any such model will be L_α for some countable ordinal α. By minimizing α, we see in this case that there is a smallest transitive model $L_\alpha \models \text{ZFC}$. Let c be the L-code for this minimal α. Note that $\{\, c \,\}$ is decidable, since on input x, we can check whether it is an L-code for an ordinal ξ such that $L_\xi \models \text{ZFC}$, and if so, whether ξ is the smallest such ordinal. If so, it must be that $\xi = \alpha$ and $x = c$. From c, we may compute a relation E on $\mathbb{N}$ such that $\langle L_\alpha, \in\rangle \cong \langle \mathbb{N}, E\rangle$. Let M be the collection of pairs $\langle c, n\rangle$, where $n \in \mathbb{N}$. This is a decidable set, because $\{\, c \,\}$ is decidable. The idea is that $\langle c, n\rangle$ represents the set coded by n with respect to E. Define $\langle c, n\rangle \;\bar{E}\; \langle c, m\rangle$ if $n \; E \; m$. This is a computable relation, because E is c-computable and $\{c\}$ is decidable. Clearly, $\langle M, \bar{E}\rangle$ is isomorphic to $\langle \mathbb{N}, E\rangle$, which is isomorphic to $\langle L_\alpha, \in\rangle$. So we have a computable presentation of $\langle L_\alpha, \in\rangle$.

Let us point out that this presentation is nearly decidable, in that we can decide $\mathcal{M} \models \varphi[x_1, \ldots, x_n]$ on input $\ulcorner\varphi\urcorner, x_1, \ldots, x_n$, provided $n \geq 1$. Specifically, from x_1 we can compute the real c, and from c we can effectively enumerate the whole structure $\langle M, E\rangle$. Having done so, we can compute whether $\mathcal{M} \models \varphi[x_1, \ldots, x_n]$ according to the Tarskian definition of truth. This method makes fundamental use of the information c that is present in any of the parameters, so it does not help us to decide whether a given sentence holds in $\mathcal{M}$, if we are not given such a parameter.

To make the model fully decidable, therefore, we assume $\ulcorner\in\urcorner = c$. Since $\{c\}$ is decidable, this language remains decidable (although no longer enumerable in any nice sense). The point now is that if we are given a sentence σ, and the symbol $\in$ appears in it, then we can compute the real c from $\ulcorner\in\urcorner$ and thereby once again effectively enumerate the whole structure $\mathcal{M}$, allowing us to compute whether σ holds. If $\in$ does not occur in σ, then σ is an assertion in the language of equality, which either holds or fails in all infinite models, and we can computably determine this in $\omega + 1$ many steps. □

If one allows $\ulcorner\in\urcorner = c$, then one can actually take the underlying set of $\mathcal{M}$ to be $\mathbb{N}$, since if one has already coded c into the language, there is no additional need to code c into the individual elements of $\mathcal{M}$. In this case, one has a decidable presentation of the form $\langle \mathbb{N}, E\rangle$. We caution in this case that the relation E is not computable, but only computable relative to c. This does not prevent the

model from being a computable model, however, since in order to be a computable model, the relations need only be computable from their Gödel codes. This may be considered to be a quirk in the definition of computable model, but in order to allow for uncountable languages, we cannot insist that the relations of a computable model are individually computable, but rather only computable from their Gödel codes.

Similar arguments establish:

Theorem 50. *If there is a transitive model of* ZFC *with an inaccessible cardinal (or a Mahlo cardinal or ω^2 many weakly compact cardinals, etc.), then the smallest such model has an infinite time decidable presentation.*

Proof. If there is a transitive model of ZFC plus any of these large cardinal hypotheses, then there is one satisfying $V = L$. Hence, as argued in Theorem 48, the theory holds in some countable L_α. By using the L-code c of the minimal such model, we can build a decidable presentation as above. □

If one wants to consider set theoretic theories inconsistent with $V = L$, then a bit more care is needed.

Theorem 51. *If there is a transitive model of* ZFC*, then there is transitive a model of* ZFC + ¬CH *with an infinite time decidable computable presentation.*

Proof. Let L_α be the minimal transitive model of ZFC. This is a countable transitive model, and so there is an L-least set G in L such that G is L_α-generic for the forcing $\mathrm{Add}(\omega, \omega_2)^{L_\alpha}$. Thus, $L_\alpha[G] \models \mathrm{ZFC} + \neg\mathrm{CH}$. The set G appears in some countable L_β, where $\alpha < \beta < \omega_1$. Let d be the L-code of the pair $\langle \alpha, \beta \rangle$. Thus, $\{\, d \,\}$ is decidable, because given any real z, we can check if z is an L-code for a pair $\langle \alpha', \beta' \rangle$ such that $L_{\alpha'}$ is the smallest model of ZFC and β' is smallest such that $L_{\beta'}$ has an $L_{\alpha'}$-generic filter G for $\mathrm{Add}(\omega, \omega_2)^{L_{\alpha'}}$. Using the real d, we can compute a relation E on $\mathbb{N}$ such that $\langle L_\alpha[G], \in \rangle \cong \langle \mathbb{N}, E \rangle$. Let N be the set of pairs $\langle d, n \rangle$ where $n \in \mathbb{N}$, and define $\langle d, n \rangle \; E \; \langle d, m \rangle$ if $n \; E \; m$. Again, this structure is computable, and it is isomorphic to $\langle L_\alpha[G], \in \rangle$, as desired. By taking $\ulcorner \in \urcorner = d$, the model is decidable as in Theorem 49. □

Clearly this method is very flexible; it provides decidable presentations of transitive models of any theory having a transitive model in L.

8. Future Directions

We close this paper by mentioning a number of topics for future research.

Infinitary languages $L_{\omega_1,\omega}$. In the context of infinite time computable model theory, it is very natural to consider infinitary languages, which are still easily coded into the reals. With any writable structure or for a structure whose domain we can search, one can still compute the Tarskian satisfaction relation. What other examples and phenomenon exist here?

Infinite time computable equivalence relation theory. The idea is to investigate the analogue of the theory of Borel equivalence relations under Borel reducibility. Here, one wants to consider infinite time computable reductions. Some of these issues are present already in our analysis of the computable quotient presentation problem in Section 5 and particularly Theorem 38. How much of the structure of Borel equivalence relations translates to the infinite time computable context?

Infinite time computable cardinalities. The computable cardinalities are the equivalence classes of the decidable sets by the computable equinumerousity relation. What is the structure of the computable cardinalities?

Infinite time computable Lowenheim-Skolem Theorems. While Theorem 28 shows that the infinite time computable upward Lowenheim-Skolem theorem holds in L, our analysis leaves open the question of whether it is consistent with ZFC that there could be a decidable countable model having no size continuum decidable elementary extension. If so, the infinite time computable upward Lowenheim-Skolem theorem will be independent of ZFC. In addition, our analysis does not fully settle the infinite time computable downward Lowenheim-Skolem theorem.

References

[DHS05] Vinay Deolalikar, Joel David Hamkins, and Ralf-Dieter Schindler. $P \neq NP \cap \text{co-}NP$ for infinite time turing machines. *Journal of Logic and Computation*, 15(5):577–592, October 2005.

[EGNR98] Yuri L. Ershov, Sergey S. Goncharov, Anil Nerode, and Jeffrey B. Remmel, editors. *Handbook of Recursive Mathematics, Volume 1: Recursive Model Theory*, volume 138 of *Studies in Logic and the Foundations of Mathematics*. Elsevier, 1998.

[Ham02] Joel David Hamkins. Infinite time turing machines. *Minds and Machines*, 12(4):521–539, 2002. (special issue devoted to hypercomputation).

[Ham05] Joel David Hamkins. Infinitary computability with infinite time Turing machines. In Barry S. Cooper and Benedikt Löwe, editors, *New Computational Paradigms*, volume 3526 of *LNCS*, Amsterdam, June 8-12 2005. CiE, Springer-Verlag.

[HL00] Joel David Hamkins and Andy Lewis. Infinite time Turing machines. *J. Symbolic Logic*, 65(2):567–604, 2000.

[HL02] Joel David Hamkins and Andy Lewis. Post's problem for supertasks has both positive and negative solutions. *Archive for Mathematical Logic*, 41(6):507–523, 2002.

[HS01] Joel David Hamkins and Daniel Seabold. Infinite time Turing machines with only one tape. *Mathematical Logic Quarterly*, 47(2):271–287, 2001.

[HW03] Joel David Hamkins and Philip Welch. $P^f \neq NP^f$ for almost all f. *Mathematical Logic Quarterly*, 49(5):536–540, 2003.

[Jec03] Thomas Jech. *Set Theory*. Spring Monographs in Mathematics. Springer, 3rd edition, 2003.

[Koe05] Peter Koepke. Turing computations on ordinals. *Bulletin of Symbolic Logic*, 11(3):377–397, September 2005.

[LÖ1] Benedikt Löwe. Revision sequences and computers with an infinite amount of time. *Logic Comput.*, 11(1):25–40, 2001.

[Wel00a] Philip Welch. Eventually infinite time Turing machine degrees: Infinite time decidable reals. *Journal of Symbolic Logic*, 65(3):1193–1203, 2000.

[Wel00b] Philip Welch. The lengths of infinite time Turing machine computations. *Bulletin of the London Mathematical Society*, 32(2):129–136, 2000.

[Wel05] Philip Welch. The transfinite action of 1 tape Turing machines. In Barry S. Cooper and Benedikt Löwe, editors, *New Computational Paradigms*, volume 3526 of *LNCS*, Amsterdam, June 8-12 2005. CiE, Springer-Verlag.

J. D. Hamkins, The College of Staten Island of The City University of New York, Mathematics, 2800 Victory Boulevard, Staten Island, NY 10314 & The Graduate Center of The City University of New York, Ph.D. Program in Mathematics, 365 Fifth Avenue, New York, NY 10016

E-mail address: jhamkins@gc.cuny.edu, http://jdh.hamkins.org

R. Miller, Queens College of The City University of New York, Mathematics, 65-30 Kissena Boulevard, Flushing, New York 11367 & The Graduate Center of The City University of New York, Ph.D. Program in Computer Science, 365 Fifth Avenue, New York, NY 10016

E-mail address: rmiller@forbin.qc.edu

D. Seabold, Department of Mathematics, Hofstra University, Hempstead, NY 11549-1030

E-mail address: matdes@hofstra.edu

S. Warner, Department of Mathematics, Hofstra University, Hempstead, NY 11549-1030

E-mail address: matsjw@hofstra.edu