Koepke Machines and Satisfiability for Infinitary Propositional Languages

Merlin Carl[1,2], Benedikt Löwe[3,4,5(✉)], and Benjamin G. Rin[6]

[1] Fachbereich Mathematik und Statistik,
Universität Konstanz, 78457 Konstanz, Germany
merlin.carl@uni-konstanz.de
[2] Fakultät für Informatik und Mathematik,
Universität Passau, Innstraße 33, 94032 Passau, Germany
[3] Institute for Logic, Language and Computation, Universiteit van Amsterdam,
Postbus 94242, 1090GE Amsterdam, The Netherlands
b.loewe@uva.nl
[4] Fachbereich Mathematik, Universität Hamburg,
Bundesstrasse 55, 20146 Hamburg, Germany
[5] Christ's College, Churchill College, and Faculty of Mathematics,
University of Cambridge, Wilberforce Road, Cambridge CB3 0WA, England
[6] Departement Filosofie En Religiewetenschap, Universiteit Utrecht,
Janskerkhof 13, 3512BL Utrecht, The Netherlands
b.g.rin@uu.nl

Abstract. We consider complexity theory for Koepke machines, also known as Ordinal Turing Machines (OTMs), and define infinitary complexity classes ∞-**P** and ∞-**NP** and the OTM analogue of the *satisfiability problem*, denoted by ∞-SAT. We show that ∞-SAT is in ∞-**NP** and ∞-**NP**-hard (i.e., the problem is ∞-**NP**-complete), but not OTM decidable.

1 Infinitary Computation and Its Running Times

1.1 Introduction

Various versions of Turing machines for infinitary computation have been proposed. They all have in common that they have ordinal-indexed tapes on which they can read and write symbols from a finite alphabet Σ, they run in ordinal-indexed steps of time, and follow the usual instructions for Turing machines for the successor ordinal steps. The first such type of machines were the *Hamkins-Kidder machines* or *Infinite Time Turing Machines* (ITTMs) defined in [5]. These machines have a regular tape of order type ω, but do not have to halt in finite time: instead, they can run through transfinite ordinal time steps. This results in an asymmetric situation between time and space as an ITTM can run through the class of all ordinals, but only has ω many cells to write on. In [10,11], Koepke symmetrised ITTMs and defined what are now known as *Koepke machines* or *Ordinal Turing Machines* (OTMs): OTMs have a class-sized tape indexed by ordinals and run through ordinal time. Other machine

J. Kari et al. (Eds.): CiE 2017, LNCS 10307, pp. 187–197, 2017.
DOI: 10.1007/978-3-319-58741-7_19

concepts include machines that restrict the space to a given ordinal α but run through arbitrary ordinal time and machines where both time and space are symmetrically restricted to an ordinal α (cf. [3,12,14]).

The symmetry between space and time for Koepke machines reflects that of finitary Turing machines. The first author has argued in [1] that Koepke machines are the natural infinitary analogue for finitary computability theory. In this paper, we shall study deterministic and nondeterministic polynomial time computation for Koepke machines. In Sect. 1.2, we give the basic definitions of our model of computation and its running time analysis. Complexity theory for infinitary computation was introduced by Schindler in [15] in the context of Hamkins-Kidder machines; for Koepke machines, the definitions were discussed by the second author and Winter [13,16,17]. We give precise definitions in this tradition (introducing the complexity class ∞-**NP**) and discuss fundamental differences between finitary and infinitary computation in Sect. 2. In Sect. 3, we introduce the OTM analogue of the satisfiability problem ∞-SAT, show that it is in ∞-**NP** and that every problem in ∞-**NP** polynomially reduces to it (i.e., ∞-SAT is ∞-**NP**-hard). However, due to the phenomena discussed in Sect. 2.2, being ∞-**NP**-complete does not necessarily imply that ∞-SAT is OTM decidable: in Sect. 4, we show that it is not (and discuss a notable difference between the general decision problem ∞-SAT and its countable fragment).

1.2 Basic Definitions

In the following, we shall be working with Koepke machines, or OTMs, sometimes allowing ordinal parameters in our computations. For detailed definitions, we refer the reader to [10].

An *OTM input* is a function $X : \alpha \to \Sigma$ where $\mathrm{lh}(X) := \alpha$ is an ordinal called the *length of* X. We assume that the tape alphabet Σ contains a blank symbol $\llcorner$ that allows us to express shorter OTM inputs as longer ones: if $\mathrm{lh}(X) = \alpha < \beta$, we consider $X^* : \beta \to \Sigma$ with $X^*(\gamma) := X(\gamma)$ for $\gamma < \alpha$ and $X^*(\gamma) = \llcorner$ for $\gamma \geq \alpha$ and identify X and X^*. A class of OTM inputs is called an *OTM decision problem*. A Koepke machine M *decides an OTM decision problem* C *in parameter* η if it halts with parameter η for every OTM input X and outputs 1 if and only if $X \in C$; an OTM decision problem is called *OTM decidable in parameter* η if there is a Koepke machine deciding it in parameter η. Parameter-free computation is the special case where η is a recursive ordinal (e.g., 0). An OTM decision problem is called *bounded with bound* λ if for every OTM input $X \in C$, we have that $\mathrm{lh}(X) < \lambda$.

We emphasize that (as in the finitary case) most natural decision problems do not occur as classes of OTM inputs, but have to be formally encoded as OTM decision problems. Typically, they come in the form of some class D of objects of some particular kind (e.g., formulas, trees, graphs, etc.) together with a coding (class) function code such that for every (relevant) set Z, $\mathsf{code}(Z)$ is an OTM input, and $C := \{\mathsf{code}(Z)\,;\, Z \in D\}$ is an OTM decision problem. We shall see below that the choice of coding is crucial in the infinitary case.

Let $f : \mathrm{Ord} \to \mathrm{Ord}$ be an increasing class function (in the following, we shall refer to these as *complexity functions*). A class function f is a *polynomial function* if there are ordinals $\alpha_n \geq ... \geq \alpha_0$ such that for all γ, we have

$$f(\gamma) = \gamma^{\alpha_n} + \gamma^{\alpha_n - 1} + ... + \gamma^{\alpha_0}.$$

We say that a Koepke machine M is a *time f machine* if the machine halts for every OTM input X in less than $f(\mathrm{lh}(X))$ steps.

If C is an OTM decision problem, we write $C \in \mathbf{Time}(f)$ if there is a time f machine deciding C and $C \in \infty\text{-}\mathbf{P}$ if there is a polynomial function f such that $C \in \mathbf{Time}(f)$. In the latter case, we say that it is *OTM polynomial time decidable*.

Similarly, if C and D are OTM decision problems, we say that *C is reducible to D in time f* if there is a time f machine that takes an OTM input X and produces an output Y such that $X \in C$ if and only if $Y \in D$. We say that *C is reducible to D in polynomial time* if there is a polynomial function f such that C is reducible to D in time f.

Proposition 1. *If C is a bounded OTM decision problem with bound $\lambda \geq \omega$, then exactly one of the following holds:*

(i) The problem C is not OTM decidable in parameter λ; or
(ii) the problem C is decided by a time c machine in parameter λ, where c is the constant function $\alpha \mapsto |\lambda|^+$.

Proof. Suppose that C is decidable by a Koepke machine M in parameter λ. In particular, for every input X, the machine M halts. If $\mathrm{lh}(X) < \lambda$, a standard Löwenheim-Skolem argument using the absoluteness of computations shows that there is some $\alpha < |\lambda|^+$ such that the model $\mathbf{L}_\alpha[X]$ is a model of "the computation of M with input X and parameter λ halts". But then the computation must halt before $\alpha < |\lambda|^+$. Checking whether $\mathrm{lh}(X) < \lambda$ can be done in time λ using the parameter λ. □

The proof of Proposition 1 has two immediate consequences for running times of infinitary computations:

First, while Koepke machines can in principle use the entire length of the class of ordinals for their computation time, halting Koepke machines (and these are the only ones that matter for decision problems) do not: they can never substantially outrun the size of the input (in the sense that, if the input has cardinality κ, then the computation will take less than $(\kappa^+)^{\mathbf{L}}$ many steps). This is a marked difference to the finitary case.

This immediately implies that the relevant operations for running time analysis of Koepke machines must necessarily be *ordinal operations* rather than *cardinal operations* since any non-trivial cardinal operations on infinite ordinals will move beyond the bounds of Proposition 1.

This in turn yields a second important consequence: as in the finitary case, infinitary complexity theory is sensitive to the encoding of the input; the more

efficient the input encoding is, the harder it is to prove complexity bounds for a decision problem. But every ordinal $\kappa \leq \alpha \leq \kappa^+$ can be encoded by a set of order type κ, so there is a maximally efficient encoding in terms of input length.

We illustrate this phenomenon by showing that any OTM decision problem that is decidable in time f by a machine M has a re-coded version such that the re-coded version of M is not a time f machine. Let κ be a cardinal and C be an OTM decision problem that was obtained from some class D by means of a coding function code such that $C = \{\mathsf{code}(Z)\,;\, Z \in D\}$. Let $f : \mathrm{Ord} \to \mathrm{Ord}$ be a complexity function and M a time f machine that decides C and for the sake of non-triviality, assume that $f(\kappa) < \kappa^+$. Now, if there is some OTM input $X = \mathsf{code}(Z)$ with $\mathrm{lh}(X) < \kappa^+$ such that M takes more than $f(\kappa)$ many steps before halting. Then let $\lambda := \max\{f(\kappa), \mathrm{lh}(X)\}$, let π be a bijection between κ and λ, and let code^* be the coding function corresponding to the combination of code and π. In particular, $\mathsf{code}^*(Z)$ has length κ. If we write $C^* := \{\mathsf{code}^*(Z)\,;\, Z \in D\}$, then the appropriately re-coded version of M does not decide C^* in time f since it will run for more than $f(\mathrm{lh}(\mathsf{code}^*(Z))) = f(\kappa)$ many steps.

Consequently, in infinitary computation, a sufficiently efficient coding for the input can potentially destroy the complexity properties of any machine. Hence, we need to assume that the encoding of the decision problem respects the *natural length* of the objects being coded.

2 Complexity Theory for Koepke Machines

2.1 Definitions

Nondeterministic complexity classes for infinitary computation were first introduced by Schindler in [15] for Hamkins-Kidder machines; Schindler's definition did not use nondeterministic Hamkins-Kidder machines, but defined the class **NP** in terms of checking a witness. This was linked in [13, Proposition 11] to nondeterministic Hamkins-Kidder machines. Schindler exploited the asymmetry between time and space for Hamkins-Kidder machines and showed that for his definitions of **P** and **NP** for Hamkins-Kidder machines, we get $\mathbf{P} \subsetneqq \mathbf{NP}$. His results were later improved to $\mathbf{P} \subsetneqq \mathbf{NP} \cap \mathbf{co\text{-}NP}$ in [4]; cf. also [6,13,16,17].

In the following, we give the corresponding definitions for Koepke machines that were essentially first developed by Winter in [16]. We call a class D of pairs of OTM inputs a *witnessed OTM decision problem*. If C is an OTM decision problem and D is a witnessed OTM decision problem, we say that C *is the projection of* D if

$$X \in C \iff \exists W((W, X) \in D);$$

i.e., if $(W, X) \in D$, we interpret W as a witness for the membership of X in C.

As usual, if X is an OTM input of limit ordinal length λ, we can consider X as a pair (X_0, X_1) of OTM inputs of length λ via $X_0(\mu + n) := X(\mu + 2n)$ and $X_1(\mu + n) := X(\mu + 2n + 1)$ (for limit ordinals $\mu < \lambda$). If $f : \mathrm{Ord} \to \mathrm{Ord}$ is a

complexity function, we say that a Koepke machine M is a $*$-*time* f *machine* if the machine halts for every OTM input X in less than $f(\mathrm{lh}(X_1))$ steps.

If D is a witnessed OTM decision problem and M is a Koepke machine, we say that M *decides* D if it halts on all OTM inputs and it outputs 1 on input X if and only if $(X_0, X_1) \in D$. If C is an OTM decision problem, we write $C \in \mathbf{NTime}(f)$ if there is a witnessed OTM decision problem D such that C is the projection of D and there is a $*$-time f machine deciding D. We write $C \in \infty$-$\mathbf{NP}$ if there is a polynomial function f such that $C \in \mathbf{NTime}(f)$.

2.2 The Fundamental Difference Between Finitary and Infinitary Computation

In the case of ordinary Turing machines and complexity functions $f : \mathbb{N} \to \mathbb{N}$, we can recapture nondeterministic computation by deterministic computation: suppose that you have some complexity function $f : \mathbb{N} \to \mathbb{N}$ and some decision problem C that is decided by a $*$-time f machine M. This means that there is a witnessed decision problem D such that C is the projection of D. On input (W, X) with $\mathrm{lh}(X) = n$, the machine halts in less than $f(n)$ steps. In particular, it reads at most the first $f(n)$ many digits of W, so we can ignore the rest of the information in W. This allows us to run an *exhaustive brute force algorithm* that checks all possible witnesses: there are $\Sigma^{f(n)}$ many input sequences of length $f(n)$, so we can just run M on all of these in combination with X; if $\widehat{f}(n) := f(n) \cdot \Sigma^{f(n)} \cdot k$ for a sufficiently large $k \in \mathbb{N}$, then the brute force algorithm checks in time $\widehat{f}$ whether $X \in C$. This argument breaks down for infinitary computation, as will be shown in Proposition 2.

Since the state of a Koepke machine is absolute between transitive models of set theory, the content of the tape has to be constructible. Thus, for the discussion of brute force algorithms, it is sensible to work under the assumption of $\mathbf{V}$=$\mathbf{L}$.

In general, if M is any Koepke machine and f is any complexity function, we say that a machine $\widehat{M}$ is an *exhaustive brute force machine associated to* M *relative to* f if at input Y with $\mathrm{lh}(Y) = \alpha$, the machine $\widehat{M}$ successively writes all OTM inputs Z of length $f(\alpha)$ on the scratch tape and then runs the machine M on input X with $X_0 = Z$ and $X_1 = Y$. It gives the output 0 if all of the runs of M produced output 0 and 1 if one of the runs of M produced 1. The above argument shows for finitary computation that $\widehat{M}$ is a time $\widehat{f}$-machine if M was a $*$-time f machine.

Proposition 2. *Assume* $\mathbf{V}$=$\mathbf{L}$. *Suppose that* f *is a complexity function such that there is some* $\alpha \geq \omega$ *with* $f(\alpha) \geq |\alpha|$ *and that* M *is a* $*$*-time* f *machine and* $\widehat{M}$ *is an exhaustive brute force machine associated to* M *relative to* f. *Then* $\widehat{M}$ *does not halt for any OTM input of length* α.

Proof. Let X be an OTM input of length α. The machine $\widehat{M}$ runs for at least $2^{|\alpha|} \geq |\alpha|^+$ many steps since the exhaustive brute force machine has to produce all OTM inputs of length $f(\alpha) \geq |\alpha|$. But Proposition 1 tells us that no machine can run for $|\alpha|^+$ many steps on input X and after that still halt. □

The fact that for infinitary computation, decision problems that are nondeterministically decidable can be deterministically undecidable has been observed and used by Winter [16, p. 74]. We shall provide a concrete example for this in Sect. 4.

3 An Infinitary Analogue of SAT

In finitary computation, the decision problem SAT is the set of satisfiable propositional formulas.

We define the natural analogue of SAT in the context of Koepke machines in the style of infinitary languages (cf. [9]). Let $\mathrm{Var} \subseteq \mathbf{L}$ be a class of propositional variables. We form *formulas* of the infinitary propositional language $\mathcal{L}_{\infty,0}$ with the unary operator $\neg$ corresponding to negation and the operator $\bigwedge$ that takes a set of formulas and produces its conjunction.

1. Every element of Var is in $\mathcal{L}_{\infty,0}$.
2. If $\varphi \in \mathcal{L}_{\infty,0}$, then so is $\neg\varphi$.
3. If Φ is a set of members of $\mathcal{L}_{\infty,0}$, then $\bigwedge \Phi$ is an element of $\mathcal{L}_{\infty,0}$.

As usual, we abbreviate $\neg\bigwedge\{\neg\varphi\,;\,\varphi \in \Phi\}$ by $\bigvee\Phi$. Formulas of the language $\mathcal{L}_{\infty,0}$ naturally correspond to labelled well-founded trees (T,ℓ) where each node in T has a set of successors and ℓ is a function from the set of leaves of T to Var. We write $\mathrm{Var}_\varphi := \mathrm{ran}(\ell)$ for the set of variables occurring in φ. By a simple Mostowski collapse argument, we may assume that $\mathrm{Var}_\varphi \subseteq \mathbf{L}_\beta$ for some $\beta < |T|^+$. The width of the tree T, denoted by $\mathrm{width}(T)$ is the supremum of the cardinalities of the sets of successors of branching nodes; the height of the tree T, denoted by $\mathrm{height}(T)$ is defined by the usual recursion on the well-founded tree structure. By interpreting the leaves t of T as propositional variables $\ell(t)$, its branching nodes as infinite conjunctions, and its non-branching nodes as negations, we can identify a formula with a labelled well-founded tree. A function $v : \mathrm{dom}(v) \to \{0,1\}$ with $\mathrm{dom}(v) \subseteq \mathrm{Var}$ is called a *valuation*. As usual, if $\varphi = (T,\ell)$ is a labelled well-founded tree, any valuation v with $\mathrm{dom}(v) \supseteq \mathrm{Var}_\varphi$ uniquely extends to a map $\widehat{v} : T \to \{0,1\}$ via the following recursive definition:

1. if $t \in T$ is a leaf, then $\widehat{v}(t) := v(\ell(t))$;
2. if $t \in T$ is a non-branching node and t' is its unique successor in T, then $\widehat{v}(t) := 1 - \widehat{v}(t')$;
3. if $t \in T$ is a branching node and X is the set of its successors in T, then $\widehat{v}(t) := \min\{\widehat{v}(x)\,;\,x \in X\}$.

We define $\widehat{v}(T,\ell)$ to be the value of $\widehat{v}$ at the root of the tree T.

Definition 3. *The problem* ∞-SAT *is to decide on input* $(T,\ell) \in \mathcal{L}_{\infty,0}$ *whether there is a valuation* v *such that* $\widehat{v}(T,\ell) = 1$.

Definition 3 does not define an OTM decision problem: in order to do so, we still need to specify in which way the formula (T,ℓ) is encoded as an OTM input.

As emphasized before, it is crucial in the realm of infinitary computation that this encoding respects the natural length of the input. We shall not specify a concrete encoding here (since it does not matter for anything that follows), but insist that the encoding function has the property that

$$\mathrm{lh}(\mathsf{code}(T,\ell)) = \max(\mathrm{width}(T), \mathrm{height}(T)).$$

With this requirement, the following results are straightforward adaptations of the classical arguments showing that SAT is **NP**-complete:

Theorem 4. *The OTM decision problem* ∞-SAT *is in* ∞-**NP**.

Theorem 5. *Every problem in* ∞-**NP** *reduces in polynomial time to* ∞-SAT.

Proof. The proof largely follows the same general structure as standard textbook proofs of the finitary Cook-Levin theorem, but with an additional component to accommodate the limit stages of machine computation. □

4 The Undecidability of ∞-SAT

In Sect. 3, we proved that ∞-SAT is ∞-**NP**-complete. However, as pointed out in Sect. 2.2, in infinitary computation, being nondeterministically decidable does not imply deterministic decidability. In this section, we shall show that ∞-SAT is not OTM decidable. In fact, the decidability behaviour of ∞-SAT restricted to constructibly countable formulas is different from the general behaviour; this follows from a theorem by Jensen and Karp:

Theorem 6 (Jensen & Karp). *If* x *is a real and* α *is a limit of* x*-admissible ordinals, then* $\Sigma_1(x)$*-sentences are absolute between* $\mathbf{V}_\alpha$ *and* $\mathbf{L}_\alpha[x]$.

Proof. This is the relativised version of a theorem proved in [7, p. 162]. The relativisation is discussed in [2, Appendix]. □

Theorem 7. *Let* $\varphi = (T,\ell) \in \mathbf{L}$ *be a constructibly countable formula, i.e.,* $\mathbf{L} \models$ *"*T *is a countable tree". Then there is an* $\alpha < \omega_1^{\mathbf{L}}$ *such that exactly one of the following holds:*

1. φ *is not satisfiable, or*
2. *there is a* $v \in \mathbf{L}_\alpha$ *such that* $\widehat{v}(\varphi) = 1$.

Proof. Since φ is countable in $\mathbf{L}$, find a real c and some $\beta < \omega_1^{\mathbf{L}}$ such that $c \in \mathbf{L}_\beta$ and c encodes φ. Let α be a limit of c-admissibles above β. The sentence "there is a valuation v such that $\widehat{v}(\varphi) = 1$" is $\Sigma_1(c)$ and hence by Theorem 6 absolute between $\mathbf{V}_\alpha$ and $\mathbf{L}_\alpha[c] = \mathbf{L}_\alpha$. So, if φ is satisfiable, then a witness of this lies in $\mathbf{L}_\alpha$. □

In contrast, the conclusion of Theorem 7 is consistently false if we allow for formulas that are not countable in $\mathbf{L}$:

Theorem 8. *There is a constructible formula $\varphi \in \mathcal{L}_{\infty,0}$ such that*

1. *for all constructible valuations $v \in \mathbf{L}$, we have that $\widehat{v}(\varphi) = 0$, and*
2. *if $\omega_1^{\mathbf{L}} < \omega_1$, then there is a valuation v such that $\widehat{v}(\varphi) = 1$.*

Proof. For every $i \in \mathbb{N}$ and $\alpha < \omega_1^{\mathbf{L}}$, let $P_{i,\alpha}$ be a propositional letter. We define

$$\Phi := \bigwedge_{i\in\mathbb{N}} \bigwedge_{\alpha<\omega_1^{\mathbf{L}}} \bigwedge_{\substack{\beta<\omega_1^{\mathbf{L}} \\ \beta\neq\alpha}} \neg(P_{i,\alpha} \wedge P_{i,\beta}) \wedge \bigwedge_{\alpha\in\omega_1^{\mathbf{L}}} \bigvee_{i\in\mathbb{N}} P_{i,\alpha} \wedge \bigwedge_{i\in\mathbb{N}} \bigvee_{\alpha\in\omega_1^{\mathbf{L}}} P_{i,\alpha}.$$

If Φ is satisfiable, then there is a surjection from $\mathbb{N}$ onto $\omega_1^{\mathbf{L}}$, so clearly, Φ is satisfiable if and only if $\omega_1^{\mathbf{L}} < \omega_1$. □

The formula φ of Theorem 8 has size $\aleph_1^{\mathbf{L}}$; by Theorem 7, it is impossible to have a smaller example. Theorem 8 allows us to refine the argument of Proposition 2: in Proposition 2, it was the exhaustivity of the machine $\widehat{M}$ that did not allow it to stop (since it would run for too long); we can now see that sometimes, even writing the witness itself can be too much to ask. If C is an OTM decision problem which is the projection of a witnessed OTM decision problem D, then we say that a Koepke machine M *decides C by producing a witness in D* if it halts on every input X and outputs either 0 or some sequence Z such that $(Z, X) \in D$. The following statement is a very weak version of our later main result, Theorem 10:

Corollary 9. *If $\omega_1^{\mathbf{L}} < \omega_1$, then ∞-SAT cannot be decided by producing a witness.*

Proof. By Theorem 8 and the assumption, we have a constructible satisfiable formula φ with no constructible witness. So, any machine that decides ∞-SAT by producing a witness will write a non-constructible valuation on the tape. But no Koepke machine can produce a non-constructible output on constructible input. Contradiction! □

Theorem 10. *The OTM decision problem ∞-SAT is OTM undecidable.*

Proof. We shall describe a Koepke machine M that produces with parameter $\omega_1^{\mathbf{L}}$ on input $i \in \omega$ a formula $\Psi_i \in \mathcal{L}_{\infty,0}$ that is satisfiable if and only if the ith Koepke machine halts with parameter $\omega_1^{\mathbf{L}}$.

The proof of the theorem will then proceed by contradiction: Assume that there is a Koepke machine M' that decides ∞-SAT, then we can combine M and M' to get a Koepke machine with parameter $\omega_1^{\mathbf{L}}$ that decides the halting problem for Koepke machines with parameter $\omega_1^{\mathbf{L}}$. But such a machine cannot exist.

The main task in the proof is the construction of the formula Ψ_i; this requires coding of $\mathbf{L}$-structures. We recall that there is a sentence $\sigma \in \mathcal{L}_\in$ such that for any transitive N, we have $(N, \in) \models \sigma$ if and only if $N = \mathbf{L}_\gamma$ for some limit ordinal γ (cf., e.g., [8, Theorem 3.3]).

For every $\beta, \gamma < \omega_1^{\mathbf{L}}$, we fix a propositional variable $P_{\beta,\gamma}$. If v is a valuation, we can define a binary relation E_v on $\omega_1^{\mathbf{L}}$ by

$$E_v := \{(\beta, \gamma)\,;\, v(P_{\beta,\gamma}) = 1\}$$

and prove the following translation from first-order logic into infinitary logic:

Lemma 11. *If $\varphi \in \mathcal{L}_{\in}$, then there is $\Phi_\varphi \in \mathcal{L}_{\infty,0}$ such that for every valuation v, we have $\widehat{v}(\Phi_\varphi) = 1$ if and only if $(\omega_1^{\mathbf{L}}, E_v) \models \varphi$.*

Proof. This is an easy induction on the formula complexity of φ. The induction steps are trivial for propositional connectives. Concerning the quantifiers, we use infinite conjunctions to express universal quantifiers and infinite disjunctions for existential quantifiers in the obvious way. □

We now add additional propositional variables $B_{\beta,\gamma}$ for every $\beta, \gamma < \omega_1^{\mathbf{L}}$ to encode an embedding from $(\omega_1^{\mathbf{L}} + 1, \in)$ into $(\omega_1^{\mathbf{L}}, E_v)$. If v is any valuation, we define a second binary relation

$$\pi_v := \{(\beta, \gamma)\,;\, v(B_{\beta,\gamma}) = 1\}$$

similar to E_v, but based on the values of $B_{\beta,\gamma}$ instead of the values of $P_{\beta,\gamma}$.

Lemma 12. *There is a formula Ξ such that for all valuations v, we have have $\hat{v}(\Xi) = 1$ if and only if π_v is a structure-preserving embedding from $(\omega_1^{\mathbf{L}} + 1, \in)$ into $(\omega_1^{\mathbf{L}}, E_v)$.*

Proof. Similar to the formula in the proof of Theorem 8. □

We write ψ_i for the formula expressing "the ith Koepke machine with parameter $\omega_1^{\mathbf{L}}$ halts" and notice that this is first-order expressible in all structures $\mathbf{L}_\eta$ for $\eta > \omega_1^{\mathbf{L}}$ (by condensation, in any such $\mathbf{L}_\eta$, the ordinal $\omega_1^{\mathbf{L}}$ is definable as the smallest ordinal that does not have a real coding it).

Let $S := \{s \in \mathbf{L}\,;\, s : \omega \to \omega_1^{\mathbf{L}}\}$. An easy condensation argument shows that $S \subseteq \mathbf{L}_{\omega_1^{\mathbf{L}}}$. We now write

$$\Psi_i := \Xi \wedge \Phi_{\sigma \wedge \psi_i} \wedge \bigwedge_{s \in S} \bigvee_{i \in \omega} \neg P_{s(i+1),s(i)}.$$

Lemma 13. *The formula Ψ_i is satisfiable if and only if the ith Koepke machine with parameter $\omega_1^{\mathbf{L}}$ halts.*

Proof. "⇐". If the ith Koepke machine with parameter $\omega_1^{\mathbf{L}}$ halts, then by the proof of Proposition 1, it has halted at some time $\omega_1^{\mathbf{L}} < \eta < \omega_2^{\mathbf{L}}$. Find a bijection $j : \omega_1^{\mathbf{L}} \to \mathbf{L}_\eta$. Since $\eta > \omega_1^{\mathbf{L}}$, we find $\gamma_\beta \in \omega_1^{\mathbf{L}}$ such that $j(\gamma_\beta) = \beta$ for every $\beta < \omega_1^{\mathbf{L}}+1$. We define a valuation as follows: $v(P_{\beta,\gamma}) = 1$ if and only if $j(\beta) \in j(\gamma)$ and $v(B_{\beta,\gamma}) = 1$ if and only if $\gamma = \gamma_\beta$. It is easy to check that $\widehat{v}(\Psi_i) = 1$.

"⇒". If $\widehat{v}(\Psi_i) = 1$, then define E_v and π_v as above. The structure $(\omega_1^{\mathbf{L}}, E_v)$ satisfies $\sigma \wedge \psi_i$ by Lemma 11 and is well-founded by $\bigwedge_{s\in S} \bigvee_{i\in\omega} \neg P_{s(i+1),s(i)}$ (there are no descending E_v-sequences). So by Mostowski's Collapsing Lemma, it is isomorphic to a transitive structure $(N, \in) \models \sigma \wedge \psi_i$. This means that $N = \mathbf{L}_\eta$ for some limit ordinal η. But Lemma 12 shows that $\omega_1^{\mathbf{L}} + 1$ embeds into $\mathbf{L}_\eta$, and hence, $\eta > \omega_1^{\mathbf{L}}$. Therefore, $\mathbf{L}_\eta$ sees that the ith Koepke machine with parameter $\omega_1^{\mathbf{L}}$ halts. Now the claim follows from absoluteness. □

Clearly, there is a Koepke machine that produces, with parameter $\omega_1^{\mathbf{L}}$, upon input $i \in \omega$, the formula Ψ_i. As mentioned before, this finishes the proof of Theorem 10 by contradiction. □

We note that the proof of Theorem 10 can be generalised to show that there is no ordinal α such that ∞-SAT is OTM decidable in the ordinal parameter α.

We also mention that it is possible to define OTM analogues of other classical **NP**-complete problems such as 3SAT and Subset Sum and prove their ∞-**NP**-completeness as well as an OTM analogue of Ladner's theorem ("there are OTM decision problems that are in ∞-**NP**, but neither in ∞-**P** nor ∞-**NP**-complete"). We shall describe these results in future work.

References

1. Carl, M.: Towards a Church-Turing-Thesis for infinitary computation (2013) preprint. arXiv:1307.6599
2. Carl, M.: Infinite time recognizability from random oracles and the recognizable jump operator. Computability (to appear)
3. Dawson, B.: Ordinal time Turing Computation. Ph.D. thesis, University of Bristol (2009)
4. Deolalikar, V., Hamkins, J.D., Schindler, R.: $\mathbf{P} \neq \mathbf{NP} \cap \mathbf{co\text{-}NP}$ for infinite time Turing machines. J. Log. Comput. **15**(5), 577–592 (2005)
5. Hamkins, J.D., Lewis, A.: Infinite time turing machines. J. Symb. Log. **65**(2), 567–604 (2000)
6. Hamkins, J.D., Welch, P.D.: $\mathbf{P}^f \neq \mathbf{NP}^f$ for almost all f. Math. Log. Q. **49**(5), 536–540 (2003)
7. Jensen, R.B., Karp, C.: Primitive recursive set functions. In: Axiomatic Set Theory. Proceedings of the Symposium in Pure Mathematics of the American Mathematical Society held at the University of California, Los Angeles, California, 10 July–5 August, vol. XIII/I of Proceedings of Symposia in Pure Mathematics, pp. 143–176. American Mathematical Society (1971)
8. Kanamori, A.: The Higher Infinite. Large Cardinals in Set Theory from Their Beginnings. Springer Monographs in Mathematics, 2nd edn. Springer, Heidelberg (2003)
9. Karp, C.: Languages with Expressions of Infinite Length. North-Holland, Amsterdam (1964)
10. Koepke, P.: Turing computations on ordinals. Bull. Symb. Log. **11**(3), 377–397 (2005)
11. Koepke, P.: Ordinal computability. In: Ambos-Spies, K., Löwe, B., Merkle, W. (eds.) CiE 2009. LNCS, vol. 5635, pp. 280–289. Springer, Heidelberg (2009). doi:10.1007/978-3-642-03073-4_29

12. Koepke, P., Seyfferth, B.: Ordinal machines and admissible recursion theory. Ann. Pure Appl. Log. **160**, 310–318 (2009)
13. Löwe, B.: Space bounds for infinitary computation. In: Beckmann, A., Berger, U., Löwe, B., Tucker, J.V. (eds.) CiE 2006. LNCS, vol. 3988, pp. 319–329. Springer, Heidelberg (2006). doi:10.1007/11780342_34
14. Rin, B.: The computational strengths of α-tape infinite time turing machines. Ann. Pure Appl. Log. **165**(9), 1501–1511 (2014)
15. Schindler, R.: $\mathbf{P} \neq \mathbf{NP}$ infinite time turing machines. Monatsh. Math. **139**, 335–340 (2003)
16. Winter, J.: Space complexity in infinite time Turing machines. Master's thesis, Universiteit van Amsterdam. ILLC Publications MoL-2007-14 (2007)
17. Winter, J.: Is $\mathbf{P} = \mathbf{PSPACE}$ for Infinite time turing machines? In: Archibald, M., Brattka, V., Goranko, V., Löwe, B. (eds.) ILC 2007. LNCS, vol. 5489, pp. 126–137. Springer, Heidelberg (2009). doi:10.1007/978-3-642-03092-5_10