On linear order and computation

The expressiveness of interactive computations on linear orders and computations indexed by ordinals

Ryan Bissell-Siders
Department of Mathematics and Statistics
Faculty of Science
University of Helsinki
Finland

Academic dissertation
To be presented, with the permission of the Faculty of Science of the University of Helsinki, for public criticism in CK112 Exactum
on the 21st *of November, 2008 at noon.*

ACKNOWLEDGEMENTS

I would like to thank my academic supervisor, Professor Jouko Väänänen, whose lectures books and articles always inspire me. I am grateful, too, to the logic group at the University of Helsinki, which was an exciting place to study these past years because of the thoughtful faculty and my motivated peers.

For financial support, many thanks to MALJA for their graduate school grant and for travel funding, to the University's Science Foundation (Tiedesäätio) for funding a year-long research project, to the Department of Mathematics and Statistics for the teaching experience (and salary!), and to the chancellor for support while writing and publishing the thesis, and for funding to attend conferences.

Warmest thanks to my family for their patience and curiosity, and especially to my parents for their support and encouragement and everything they have given, from the first lessons to the last welcome home.

CONTENTS

1. INTRODUCTION

This dissertation studies how the Dynamic Ehrenfeucht-Fraïssé Game computes queries on ordered sets and how a finite number of automata running over ordinals in ordinal time computes queries about truth in the universe of constructible sets. In a sense, games are interactive computation. So we compute about linear orders, then once we are comfortable knowing that little can be said about a linear order, we consider what can be said about computation along a linear order. The thesis contains five papers:

I *On quantifier-rank equivalence between linear orders*, Submitted, accepted pending revisions' review, Information and Computation, WOLLIC2007 post-conference volume. I was proud to hear that something like 20% of submissions to WOLLIC were accepted, and half of those invited for post-conference publication. The paper changed a lot after the conference because I was preparing it for this thesis.

II *Dynamic Ehrenfeucht-Fraisse games on linear orders in first order and infinitary logic.* This is a manuscript.

III *Minimality considerations for ordinal computers modeling constructibility*, with Peter Koepke, Theoretical Computer Science, 394, 197-207, 2008.

IV *Register computations on ordinals*, with Peter Koepke, Archive for Mathematical Logic, 47, 529-548, 6, September 2008.

V *Ehrenfeucht-Fraïssé Games on Linear Orders*, in *Logic, Language, Information and Computation*, Lecture Notes in Computer Science, 4576, 72-82, July 2007.

These papers grew out of questions about ordinal arithmetic: definitions and computable enumerations. In 1996, I wondered

- what questions can be resolved by running a few linked pointers (automata which can alter each others' states, and sense when they collide) continuously, or monotonically, through the ordinals?

I came to Finland hoping to work on infinitary logic and I was well satisfied. Professor Väänänen assigned me a project on ordinals –

- interpreting $\equiv_k$ classes of ordinals with the same quantifier-rank-k theory as classes of ordinals with a common tail in their Cantor Normal Form expressions, and
- playing Ehrenfeucht-Fräissé games on ordinals with additional structure, such as ordinal arithmetic.

The results showed how very little can be defined, even in infinitary logic – only the first and last occurrence of each quantifier-rank-k-type. Out of this work we get a normal form for a formula of the theory of linear order in first-order and infinitary logic. All of the work for this thesis was researched and written while I was a student at the University of Helsinki. The second paper was written first, in 2004. The first paper came next, in 2005. The 3rd and 4th papers grew out of a collaboration with Professor Koepke in 2005. They answer the question posed earlier in this paragraph (the first item/bullet) which I had studied in 1996. The linear consistency game which now runs through the first paper was written into it in 2007, though it was present as the framework I was thinking in ever since I learned the model existence game in Professor Väänänen's 2003 course on infinitary languages.

We review some early work on logics with a long semantic game, abstract recursion, and linear order to put this thesis in a historical setting:

Tarski [T35] proved the decidability of some types of linear order after having considered them during a 1926-28 seminar. Turing defined the mechanical definition of definability or computability which is ubiquitous today. He used it first [T39] to investigate the question: would a tower of logics, each resolving the truth predicate of its predecessors, be complete? Gödel [G40] seems to have answered this question in the affirmative with the constructible universe. Modern treatments of this model of set theory carry out the axioms of comprehension transfinitely, and phrases like "Silver machines" suggest that theorists implicitly view this as a mechanical process by which all queries can be computed and answered in infinite time.

Tarski proved the decidability of the theory of ordinals. Kreisel [K53] asked whether a broader class of linear orders might have a decidable theory. Ehrenfeucht [E59] remarked that the consequences of a theory are decidable just in case there is an algorithm to produce, for each consistent extension of the theory, a model of that extension. Thus, the difficulty in proving the theory of linear order to be consistent, prima facia, was the great variety of models which might have to be constructed (by a single program).

Fefferman and Vaught [FV59] showed how the theory of a whole model can be derived from the theory of its summands or multiplicands – from its parts. This is very useful in aligning a whole model to satisfy a particular theory. In this thesis we will use only the shallowest notion of addition on theories of linear order, addition on sets of pairs of theories, and addition and multiplication on semi-models. Meanwhile, Tarski [T58] remarked that $L_{\omega_1\omega_1}$ defines wellorders by asserting the existence of an infinite descending chain, whereas in $L_{\omega_1\omega}$ wellorder is an axiom schema, as in $L_{\omega\omega}$. Ehrenfeucht [E60] addressed the question of deciding ordinal arithmetic with a paper interpreting quantifier-rank equivalence $\equiv_k$ in terms of a game in which the second player tries to construct a partial isomorphism of size k, and the first player insists that the most dissimilar elements be included in that partial isomorphism. Implicit here was the notion of satisfaction as a semantic game; the concatenation of two semantic games is the comparison game. Büchi [B60] showed that queries on linear orders can be treated as automata. It's common nowadays to think of automata in terms of the languages they accept. Büchi exploited the reverse: queries and model-theoretic invariants can often be interpreted as simple repetitive computations. Kreisel [K65] and Sacks described metarecursive sets and metarecursive

functions on the recursive ordinals.

The study of $L_{\omega_1\omega}$, constructibility, and deciding strong languages on linear orders (e.g., by interpreting them as automata), grew into fields of their own. Yet, the decidability of the theory of linear order remained unfinished. Läuchli and Leonard [LL66] proved that the theories of ordinals are not as varied as had been expected; indeed, three functions generate a family of linear orders which satisfy any consistent extension of the theory of linear order. They asked whether the theory of linear order is atomic; Amit and Shelah [AS76] proved that a set of functions can simultaneously preserve the set of finitely axiomatizable linear orders and generate models of any consistent extension of the theory of linear order. Lopez-Escobar [L68] refined the proof that wellorder is not definable in $L_{\omega_1\omega}$ into a normal form expressing quantifier-rank $\equiv_\alpha$ classes in $L_{\omega_1\omega}$(linear order) as the class of ordinals with a given Cantor Normal Form tail. Of course, a Normal Form goes in quite a different direction than proving that wellorder cannot be defined even with additional information.

The logic group in Helsinki studied logics whose truth predicate is defined by a semantic game, see [O90] [T90] and [VV04]. Whereas Ehrenfeucht's player I was allowed to point out at most k most-different elements, these games replace the finite number k by a *clock* $(P, <)$. In this thesis, $<$ is a linear order, but trees and posets are interesting, too. Player II must continue to create a partial isomorphism so long as the first player is able to continue diminishing the value on the clock. This notion of a semantic game has been fruitful in philosophy, as well as in mathematics. The question of whether player I would prefer to use the clock $(P, <)$ or $(Q, <)$ when comparing models $A, <$ and $B, <$ becomes complicated.

When I first planned this thesis, Professor Väänänen assigned a series of games involving ordinals with structures, especially ordinal arithmetic. In 2004 I had meant to prove something to be decidable, or decidable relative to an oracle. I still mean to publish those results, including some lovely and surprising Ehrenfeucht-Fraïssé games. In this thesis you won't find any ordinal arithmetic, because other things happened. First, I have studied a notion of infinitely repeated ordinal arithmetic – a finite number of registers moving continuously through the ordinals and reading their instructions from a single, finite program – and it turns out that this machine can compute truth in Gödel's constructible universe and effectively construct a model of set theory (and one that people care about). Explicit computation of set-theoretic constructible truth may have fruitful applications to studying interesting questions about constructible in set-theory. While studying ordinal arithmetic, I was surprised to work out many simplifications of what the literature presents as the theory of linear order; publishing them now fills out this thesis:

- A normal form for quantifier-rank classes of linear orders, which is similar to the Normal Form of [L68] for $\equiv_\alpha$ classes of ordinals, describes each $\equiv_\lambda$ class in terms of the definable ends of the linear order. This is useful for constructing models of any theory containing a linear order.

- the decidability of the theory of linear order without appeal to Ramsey's theorem,

- a simpler construction of a linear order with undecidable theory, but decidable Σ_n theories,

- a feasible enumeration of the $\equiv_4$ classes of linear order and the $\equiv_k$ classes of ordinals.
- the atomicity of the Boolean algebra of theories of linear order from a different direction than that in [AS76] (they simplify the linear orderings, we simplify the propositions),

and with a few comments about semi-models the thesis is finished and our abiding interest in ordinal arithmetic is now invisible in the following articles.

References

[T35] A. Tarski, *Grundzüge des Systemenkalküls. Erster Teil*, Fundamenta Mathematicae 25, 503-526 and *Zweiter Teil*, F.M. 26 283-301. 1935-1936.

[T39] A. Turing, *Systems of logic based on ordinals*, Proceedings of the London Mathematical Society. Second Series 45, 161–228, 1939.

[G40] K. Godel, *The Consistency of the Axiom of Choice and of the Generalized Continuum-Hypothesis with the Axioms of Set Theory*, Annals of Mathematics Studies, 3, 1940.

[T38] J. Doner, A. Mostowski, A. Tarski, *The elementary theory of well-ordering—a metamathematical study*, Logic Colloquium '77 (Proc. Conf., Wrocław, 1977), pp. 1–54, Stud. Logic Foundations Math., 96, North-Holland, Amsterdam-New York, 1978. – apparently this proof was done by quantifier elimination in the years 1938-1941.

[K53] G. Kreisel, MR0060439, *review of* Janiczak, A. *Undecidability of some simple formalized theories.* Fund. Math. 40, 131-139 (1953).

[E59] A. Ehrenfeucht *Decidability of the theory of linear ordering relation*, AMS Notices, 6.3.38:556-38, 1959.

[FV59] S. Feferman, R.L. Vaught, *The first order properties of products of algebraic systems.* Fund. Math. 47 57–103, 1959.

[T58] A. Tarski *Remarks on Predicate Logic with Infinitely Long Expressions*, Colloq. Math 6:171-176, 1958.

[E60] A. Ehrenfeucht *An application of games to the completeness problem for formalized theories* Fund. Math. 49 1960/1961 129–141.

[B60] J.R. Büchi, *Weak second order arithmetic and finite automata.*, Z. Math Log, Grundl. der Math 6:66-92, 1960.

[K65] G. Kreisel, *Model-Theoretic invariants: Applications to recursive and hyperarithmetic operations* The Theory of Models, North-Holland Publishing Company, Amsterdam, pp. 190-206, 1965.

Scott [S63] gathered completeness, interpolation, and Löwenheim-Skolem theorems for $L_{\omega_1\omega_1}$. Karp [K63] proved the undefinability of wellorder, extending a proof of Ehrenfeucht's from $L_{\omega\omega}$ to $L_{\omega_1\omega}$, and this result was sufficient to take the place of compactness in many proofs.

[S63] D. Scott *Logic with denumerably long formulas and finite strings of quantifiers*, Proceedings of the International Symposium on the Theory of Models, Berkeley 329-341, 1963.

[K63] C. Karp *Finite-Quantifier Equivalence*, Symposium on the Theory of Models, Berkeley; Published, North-Holland Publishing Co. Amsterdam, 1965 pp. 407-412, 1963.

[LL66] H. Läuchli & J. Leonard *On the elementary theory of linear order*, Fund. Math LIX:109-116, 1966.

[AS76] R. Amit, S. Shelah *The complete finitely axiomatized theories of order are dense*, Israel J. Math. 23 (1976), pp 200-208.

[L68] E.G.K. Lopez-Escobar *Well-orderings and finite quantifiers*, J. Math Society Japan, 20: 477-489, 1968.

[O90] J. Oikkonen, *Equivalence of Linear Orderings* JSL 55:1 65-73, 1990.

[T90] H. Tuuri, *Infinitary languages and Ehrenfeucht–Fraïssé games*, thesis University of Helsinki, 1990.

[VV04] J. Väänänen and Boban Velickovic, Games played on partial isomorphisms, Archive for Mathematical Logic, 43:1, 19-30, 2004.

2. SUMMARY OF CONTENTS

I *On quantifier-rank equivalence between linear orders*

A An introductory section

1 Theorem 3.0.1 gives a precise criterion for $L_{\omega_1\omega}$ quantifier-rank equivalence $\mu_0 \equiv_\alpha \mu_1$ in terms of the Cantor Normal Form of μ_i.

2 Theorem 3.0.2 gives a similar criterion for $L_{\omega_1\omega}$ or first-order quantifier-rank equivalence $\mu_0 \equiv_k \mu_1$. We don't have a Cantor Normal Form for arbitrary ordinals. What we have, instead, is an ordering of first and last occurrences in $\{x : \phi(x)\}$ which is a natural generalization of the Cantor Normal Form.
Suppose $\phi(x)$ and $\psi(x)$ are formulas of quantifier-rank $k-2$ with a single free variable, x. Then in quantifier-rank k we have sentences such as

$$\exists x(\phi(x) \wedge (\forall y(\psi(y) \to x < y)))$$

which expresses that $\inf\{x : \phi(x)\} < \inf\{x : \psi(x)\}$. By switching ϕ for ψ, $<$ for $>$, and negating the formula, the reader will see that $\equiv_\alpha$ can express the ordering on the set A of $\inf\{x : \phi(x)\}$ and $\sup\{x : \phi(x)\}$ for each formula ϕ, and whether those suprema and infima are realized. Furthermore, we could substitute for ϕ in the displayed sentence a formula of quantifier-rank $k-1$, and in this way, $\equiv_\alpha$ knows the quantifier-rank $k-1$ formulas realized between adjacent elements of A. The theorem asserts that this is all that $\equiv_\alpha$ can express. Furthermore, we can enumerate A efficiently by decomposing an arbitrary formula $\phi(x)$ into its local part and a global part, the global part is the determined by 2^α-many iterations of adding inf and sup to A.
The global part of ϕ indicates where the constant lies in A, and the local part describes the area around the constant. We form A in stages: for each cut $(b, c) : b \cup c = A, b < c$, we add to A the inf (and, symmetrically, the sup) of $\{x : \phi(x)\}$, for every *local* formula $\phi(x)$ of quantifier rank β such that:

- β is larger than the local types in some cofinal segment of b, or
- elements satisfying ϕ are not cofinal in b, and above them all is some element of b which has quantifier rank $> \beta$,

For this reason, A is closed under 2^α-many iterations of the process of adding to A the inf and sup of each $\equiv^{\text{loc}}_\beta$ class, for β the least bit in the index $\gamma < 2^\alpha$ of this closure process.

3 Theorem 3.0.3 introduces a Model Existence Game for the theory of linear orders: A sentence is consistent just in case there is a function which tells how, for any constant, we could split the linear order into two parts, and describe both parts in the same quantifier rank as the

original theory, such that the two parts add up to the original theory. The original sentence is a set of "promised types," along with the promise that these are the only types realized. If it is possible to consistently split this sentence on any of its promised types, and likewise for the "split" sentences, then all of these sentences are consistent.
If the subject matter were trees or planar graphs, rather than linear orders, we might say: "if a sentence has any model, then it has a fractal model" – that fractal model being the model created by repetitively replacing the promise of a linear order satisfying ϕ, which promises a set U of types, with, randomly for an element $x \in U$, the pair of promises ψ and ξ which split ϕ at x.

B A section on decidability and the consistency game.

1 Definition 3.1.1 defines the linear consistency game, which can prove a linear order to be consistent. Different versions of the game all work – player II stores a database of information about tne intervals, and player I can have various fields – so long as the database determines the $\equiv_\alpha$ class of each interval.

2 A winning strategy for player II witnesses the consistency of ϕ. In this way, we find that the theory of linear order is decidable, and that the decision procedure does not require any essential use of Ramsey theory, except to provide a (poor) upper bound on the length of the game, and so on the effectiveness of this procedure.

C A section on local equivalence at an element.

1 $\equiv_\alpha^{\text{loc}}$ is defined. Various definitions are equivalent – we could say that a formula $\phi(x)$ is local if the variables it refers to can't tend uniformly away from x, or we could insist that the theory between x and any variable mentioned be "small," in the sense of a metric on sentences. Maybe the best definition is in terms of games: $\mu_0 \equiv_\alpha^{\text{loc}} \mu_1$ holds just in case Player II wins the EF game in which either player can add new elements to either end of either linear order, and play there. We give a definition based on the arithmetic of linear orders, because:

2 All the useful results we need about $\equiv^{\text{loc}}$ follow quickly from the definition.

D A section on labeling $\inf\{x : \phi(x)\}$ and $\sup\{x : \phi(x)\}$, or firsts and lasts.

1 We define "cuts" and "intervals." We define a *label* to be $\inf\{x : \phi(x) \wedge b < x < c\}$ or $\sup\{x : \phi(x) : \wedge b < x < c\}$, for any cut (b, c) such that $A = b \cup c$.

2 Definition 3.3.2 spells out the conditions under which a label is definable. For finite quantifier ranks, the definable labels are a significant fraction of all labels. However, for infinite quantifier ranks, vast numbers of labels are "undefinable," and this author is grateful whenever the final ordered set A turns out to much smaller than the set of labels, rendering the theory of the linear order in that quantifier rank much shorter.

3 Definition 3.3.3 indicates how we index a sequence of closure, the union of which is an efficient enumeration of that set A of infima and suprema which $\equiv_\alpha$ can order.

4 A lemma shows that if player I preserves a "discrepancy" in A, player I can win.

5 Theorem 3.3.1 shows how player I can get a "discrepancy" in the initial game, from any difference between A or the $\equiv^{\mathrm{loc}}_{\beta}$ classes realized in any cut in A.

6 A lemma shows how the local $\equiv^{\mathrm{loc}}_{\alpha}$ types of the left end of an interval orders some labels $\inf\{x : \phi(x)\}$ and $\sup\{x : \phi(x)\}$ in the interval, and that the left and right ends of the interval label every element of A that $\equiv_{\alpha}$ orders in the interval.

7 Theorem 3.3.2 shows that the labels in A are sufficient to insure $\mu_0 \equiv_k \mu_1$ when the same labels A have the same order in μ_i and the same $\equiv^{\mathrm{loc}}_{\beta}$ classes between elements of A for every $\beta < \alpha$.

E A section called "effective decision procedures and completions" which introduces a local version of the linear consistency game. We prove that the two games are equivalent. The local game offers each player far fewer moves than the linear consistency game, and so it is possible to write down winning strategies in the local game even when the linear consistency game's game tree is prohibitively large.

1 The *local consistency game* is defined – rather than split $\equiv_k$ theories at the types they promise, we split unordered sets of promised local types into subsets.

2 Definition 3.4.2 indicates how we will "weave" the orderings implied by a series of $\equiv^{\mathrm{loc}}_{\beta}$ classes into a single ordering of a large number of constants. The result is closed under the function from (an $\equiv^{\mathrm{loc}}_{\beta}$ class τ and a subformula ϕ which τ promises) to (an $\equiv^{\mathrm{loc}}_{\beta}$ class which extends ϕ and information about how its promises are ordered with respect to the promises of τ).

3 Definition 3.4.3 and 3.4.4 define completions in case 1. the set of promised local types can be realized in a single almost locally closed set, or 2. not.

4 We define a linear order which is in the desired $\equiv_k$ class, and which is finitely axiomatizable, indeed, axiomatized by the formula in the previous two definitions.

5 Theorem: the linear order constructed in the previous definition is a model of the "piecewise dense" of definitions 3.4.3 and 3.4.4.

6 Theorem 3.4.2 shows that those formulas are complete, by finding the $\equiv_{k+m}$ class of the formula, for all m. This uses in an essential way our proof about the normal form of an $\equiv_{k+m+2}$ class – we show that the complete formula determines which $\equiv^{\mathrm{loc}}_{k+m}$ classes ϕ are realized in any of its models, by induction on m and by the theorem that an $\equiv^{\mathrm{loc}}_{k+m}$ class, as a set of $\equiv_{k+m}$ classes, only knows where some labels $\inf\{x : \psi(x)\}$ and $\sup\{x : \psi(x)\}$ occur; then we prove that the complete formula implies the ordering on the labels $\inf\{x : \phi(x)\}$ and $\sup\{x : \phi(x)\}$ for various such ϕ.

F A section about *semi-models*. A semi-model is a way of storing information, which is something like a formula, in that it is nested, and something like a model, in that its constituents are ordered, and that ordering is intended to be reflected in any model of the the semi-model.

1 We define semi-models, modifying slightly the definition of [3].

2 We prove the theorem stated (but not proved) in [3] that if semi-models provided a way of constructing a model for any consistent sentence ϕ (namely, an easy part would be to order the subformulas of ϕ, then we find a function from nodes to subtrees such that copying those subtrees below those nodes provides another semi-model of ϕ; the hard part is this: when we iterate this process infinitely, until the tree is closed (and probably infinite), how does the theory of the resulting infinite tree relate to ϕ?), then a corollary would be the decidability of the theory of liner order.

3 A lemma proves the minimal semi-models of ω, the natural numbers.

4 A lemma proves the minimal semi-models of η, the rational numbers.

5 A theorem proves that there is a semi-model for any consistent theory ϕ. An example then shows that these semi-models are very concise ways to represent ϕ – in fact, the semi-model is at once a compressed statement of ϕ and an ordered set which shows how to construct arbitrarily large semi-models of ϕ.

II *Dynamic Ehrenfeucht-Fraisse games on linear orders in first order and infinitary logic..*

A An introductory section We enumerate the theory of topologies induced by linear orders. This conveniently isolates the theory of "transitive sets of types limiting to elements." To this, the theory of linear order adds exactly the permutations on the set of types. This introduction gives a simple definition of $\equiv_k^{\text{loc}}$ which works only for topologies, but which perhaps illustrates how local types can be used in a simple setting, before we turn our attention to the theory of linear order.

1 Theorem: We use the linear consistency game to prove an enumeration of $\equiv_2$ classes of linear orders.

2 Theorem: We use the local linear consistency game to prove an enumeration of $\equiv_3^{\text{left}}$ classes of linear order.

3 A definition explain the local linear consistency game, and a theorem proves that it works. The theorem is a corollary of our work on the normal form for an $\equiv_\alpha$ theory of linear orders.

4 Theorem: the $\equiv_2^{\text{loc}}$ classes, enumerated.

5 Theorem: the sets of $\equiv_2^{\text{loc}}$ classes which appear in the local linear consistency game are enumerated, and each is shown to be inconsistent or consistent. As a result, we find an enumeration of $\equiv_3$ classes of linear orders.

B The $\equiv_4$ classes of linear order are numerous; we can estimate their number to within 1% accuracy by hand, but to resolve those 1% of cases which are difficult, we submit them to a computer program.

1 The program is defined.

2 Theorem: the program faithfully tests $\equiv_4$ classes by playing the local linear consistency game.

C Section: show that the $L_{\omega_1\omega}$ theory of linear order is decidable and we prove that the criterion we have established for finite $\equiv_k$ holds also for infinitary $\equiv_\alpha$.

1 Definition 4.3.1 decides whether labels $\inf\{x : \phi(x) \wedge b < x < c\}$ or $\inf\{x : \phi(x) \wedge b < x < c\}$ – for ϕ a local formula and $b \cup c$ a set of labels – are known to $\equiv_\alpha$.

2 Definition 4.3.2 describes the closure process which results in the set A of labels $\inf\{x : \phi(x)\}$ or $\sup\{x : \phi(x)\}$. This process runs for ordinal 2^α-many timesteps, one for each possible sequence of clocktimes in the dynamic Ehrenfeucht-Fraïssé Game of length α.

3 Lemma 4.3.1 shows that if player I preserves a "discrepancy" in A, player I can win.

4 Theorem 4.3.1 shows that player I can find that discrepancy if the ordering on A is different, or if for some $\equiv^{\mathrm{loc}}_\beta$ class ϕ and some labels $b \cup c \subseteq A$, the formula $(\phi(x) \wedge b < x < c)$ is realized in one model and not in the other.

5 Theorem: If some moves are played, an interval $[a, b]$ has been identified, and α'-many moves remain, then the fact that the same local types were realized between the same labels in the original game implies that the same holds now in the interval $[a, b]$, with reduced quantifier rank α'.

6 Theorem 4.3.3 explains our normal form for $\equiv_{\alpha+1}$, in terms of labels $\inf\{x : \phi(x)\}$ and $\sup\{x : \phi(x)\}$.

7 Theorem: as an example of the theorem, we prove a criterion for $Z \times k_0 \equiv_{\omega+m} Z \times k_1$ which can be easily verified by playing the game. However, a more interesting corollary is the following:

D Section: We prove a normal form for $L_{\omega_1\omega}$ quantifier-rank $\equiv_\alpha$ classes of ordinals – each such $\equiv_\alpha$ class defines the class of ordinals which share a certain Cantor Normal Form tail.

1 Theorem: the almost locally closed sets of ordinals are very simple. In fact, ordinals' regularity or homogeneity (away from the tail) imply that there are in fact very few locally closed sets, even fewer than the number of $\equiv^{\mathrm{loc}}$ classes, at each quantifier rank.

2 Theorem: the criterion, the same as 3.0.1, is proved as a corollary of the normal form for $L_{\omega_1\omega}$ over ordered sets.

3 We find a function from the $\equiv_k$ and $\equiv_{k-2}$ classes of ordinals which generates the $\equiv_{k+2}$ classes of ordinals. This allows:

4 Theorem: a precise enumeration of all (roughly $2^{k^{2^{\cdots}}}$) $\equiv_k$ classes of ordinals.

E Section: We create an undecidable linear order on which Σ_n is uniformly decidable.

1 The linear order is defined;

2 We give an algorithm for computing Σ_n;

3 The linear order has been defined by diagonalizing against all algorithms, so no algorithm uniformly decides its theory.

F Section: We extend the criterion to the very interesting non-wellfounded logics whose queries are computed by Dynamic Ehrenfeucht-Fraïssé Games along ordered sets which are not wellfounded (in our case, linear orders, but more generally, trees or posets).

1 Lemma: each such games is, in one sense, a wellfounded sequence of nonwellfounded games whose length is not fixed by diminishing a clock, but by a simple cardinal upper bound, since the order type of any set of moves which are strictly weaker than each other is wellfounded.

2 Theorem: replacing the tree of labels 4.3.2 of infinitary logic by a tree of labels indexed by the wellordering described in the previous lemma, we find the normal form for an $\equiv_\lambda$ quantifier-rank-λ equivalence class of linear orders.

III *Minimality considerations for ordinal computers modeling constructibility*, with Professor Peter Koepke.

A We argue that continuous computation along a linear order can model computations in which continuity is lost at a limit time.

B A continuous automaton with a finite number of registers pointing to ordinals, running over ordinal time, is defined and many of its properties are proved.

C We prove that the automaton can model the behavior of seemingly more complicated automata, by stacking information about them into its registers.

D We prove that the number of pointers required is between 4 and 17.

E We describe the speed of this automaton, when compared to an automaton which runs over ordinals and is allowed to write messages to itself at any ordinal, i.e., an infinite Time Ordinal Turing Machine.

IV *Register computations on ordinals*, with Professor Peter Koepke.

a Section: introduction

1 We describe other work on hypercomputation.

2 Our main theorem: A set of ordinals is computable (some program halts on exactly those ordinals) if and only if it is in L, Gödel's set of constructible sets.

b A section defining the ordinal register machines precisely.

1 We define the Ordinal register machines,

2 We define the process of ordinal computation,

3 We define what it means for a function from sequences of ordinals to ordinals to be computable.

c A section on algorithms

1 We prove that addition, multiplication, and composition of ordinals are ordinal computable.

2 We prove that inverses to computable functions are all computable.

d A section on Stacks of ordinals

e A section on recursion

1 We prove that anything definable by recursion is ordinal computable.

2 We write the program for reading code and executing it.

3 We prove that this stacking/simulation program faithfully reads and executes code.

f We prove that truth in the constructible universe is definable by recursion.

g We translate truth in the constructible universe into truth in the theory of sets of ordinals.

h We list axioms for the theory of Sets of Ordinals and prove that any model of that theory contains a model of ZFC.

1 Any element of ZFC is wellfounded; If R is a wellfounded collection of sets of ordinals in this notion of $\in$, then we call R a "point"; the theory of points under extension will be the model of ZFC that we build within the theory of sets of ordinals.

2 We define when two points are equivalent, so that the theory will be extensional.

3 We define a point to be "in" another point if, as wellfounded relations, the former is equivalent to a sub-ordering of the latter.

i The set of points and "in" satisfies ZFC.

1 Lemma: comprehension.

2 Theorem: the other axioms of ZFC.

V *Ehrenfeucht-Fraïssé Games on Linear Orders*

A Section: We write a finite database in which both players in the EF game can find their winning strategies.

B Section: We write semi-models for the elements of the Läuchli-Leonard hierarchy. These are very inefficient semi-models of formulas, when compared with the semi-models of item 6.

1 We define equivalence between models and semi-models.

2 We define the class of *consistent semi-models*.

3 We enumerate the Läuchli-Leonard hierarchy, i.e., $\langle 1, \times\omega, \times\omega^*, \text{finite} \sum, \sigma \rangle$, the closure of the singleton order under the four functions: take λ to $\lambda \times \omega$ or to $\lambda \times \omega^*$; take $(\lambda_i)_{i<n}$ to $\sum_{i<n} \lambda_i$ or to a dense shuffle of the λ_i.

4 We write semi-models for the results of those functions.

3. QUANTIFIER-RANK EQUIVALENCE BETWEEN LINEAR ORDERS

Abstract

We construct winning strategies for both players in the Ehrenfeucht Fraïssé Game on linear orders. To this end, we define the local quantifier-rank k theory of a linear order with a single constant, (λ, x) and prove a normal form for $\equiv_k$ classes, expressed in terms of local classes. We describe two implications of this theorem: 1. a decision procedure for whether a set U of pairs of $\equiv_k$ classes is consistent. – whether for some linear order λ, U is the set of pairs (ϕ, ψ) such that $\lambda \models \exists x(\phi^{<x} \wedge \psi^{>x})$ – which runs in time linear in the size of the formula which expresses that exactly the pairs of $\equiv_k$ classes in U are realized. The only obstacle to effectively listing the theory of linear order is the vast number of different $\equiv_k$ classes of theories of linear order. 2. a finitely axiomatizable linear order λ which we construct inside any $\equiv_k$ class of linear orders. We relate our winning strategies to semimodels of the theory of linear order. First, we situate our result in a historical background.

Introduction

That the theory of ordinals is decidable is proved in [2] and [10] and [11] as an initial step in other directions. In [4] we find the decidability of the theory of ordinals proved from a game-theoretic view. The reader of this paper must be aware of the game defined in [4]. We will not use any theorem from [4], but that paper is, in any case, the right introduction to our subject. In [6] (Theorem 3, page 411) we find infinitarily equivalent ordinals, generalizing the $\equiv_k$-equivalent ordinals of [4] (Theorem 12). On the other hand, Scott sentences of infinitary logic define any ordinal. In [8] we find smaller $\equiv_k$ ordinals than those in [6] as a result of analyzing the Ehrenfeucht-Fraïssé Game (hereafter, the EF game) to the precise solution of theorem 3.0.1:

For any ordinals μ, δ, let δ be a cutoff, with respect to which we see μ as having a *body* and *tail*: $\beta_\delta(\mu) = \{x \in \mu : x + \omega^\delta \leq \mu\}$, $\tau_\delta(\mu) = \{x \in \mu : x + \omega^\delta > \mu\}$. The separation of μ into these two pieces is useful in defining the Cantor Normal Form (hereafter CNF). Note that $\mu = \beta_\delta(\mu) + \tau_\delta(\mu)$ and ω^δ divides $\beta_\delta(\mu)$.

Theorem 3.0.1: ([8]) For any ordinals μ_0, μ_1, α, $\mu_0 \not\equiv_\alpha \mu_1$ holds just in case for some $\delta < \alpha$, one of the following holds:

1. $2 \times \delta < \alpha$ and $((\delta > 0) \wedge (\omega^\delta < \mu_0 \iff \omega^\delta < \mu_1)) \vee ((\delta = 0) \wedge (0 < \mu_0 \iff 0 < \mu_1))$.

2. $(\exists\gamma(2\times\delta<\gamma<\alpha))$ and

$$\vee_{i<2}((\beta_{\delta+1}(\mu_i)=\beta_\delta(\mu_i))\wedge(\beta_{\delta+1}(\mu_{1-i})\neq\beta_\delta(\mu_{1-i}))\wedge$$
$$((\tau_\delta(\mu_i)=\emptyset)\vee(\tau_\delta(\mu_i)\setminus\{\beta_\delta(\mu_i)\})\not\equiv_{\alpha-1}\omega^{\delta+1}+\tau_\delta(\mu_{1-i}))).$$

3. $\chi_\delta(\mu_0)\neq\chi_\delta(\mu_1)$ and $\vee_{i<2}\chi_\delta(\mu_i)<2^{\alpha-(2\times\delta)}-1$.

where $\chi_\delta(\mu_i)$ is:

- The number of x such that $\omega^\delta\times x<\mu_i$ and $\omega^\delta\times x+\omega^{\delta+1}>\mu_i$,
- $+3$ if $\omega^{\delta+1}\leq\mu_i$,
- -1 if $\delta>0$ and $\mu_i<\omega^{\delta+1}$,
- -1 if $\tau_\delta(\mu_i)\neq\emptyset$ and $(\tau_\delta(\mu_i)\setminus\{\beta_\delta(\mu_i)\})\not\equiv_{2\times\delta}\omega^\delta+\tau_\delta(\mu_i)$.

The Cantor Normal Form (hereafter CNF) of μ is similar to $\sum_\delta\omega^\delta\chi_\delta$. The first two conditions require μ_i to have the same CNF exponents δ; the final condition requires that the CNF coefficients are the same, or large. From the CNF of μ_0 and μ_1 we find a finite set of δ such that if the theorem fails at those δ, it must fail at all other δ. Theorem 3.0.1 describes the theory of ordinals precisely and can be used to imply other results. Our main theorem is, similarly, a solution to the EF game for linear orders:

Theorem 3.0.2: If λ is a linear order and k is a finite number then there is a finite function A_λ mapping *labels* into $\lambda\cup\lambda^+$ such that for any two linear orders λ and λ_0 and any clock k, $\lambda\equiv_k\lambda_0$ holds just in case A_λ and A_{λ_0} have the same labels in their domains and induce the same ordering on them, and the sequences $(Th^{\rm loc}_{k-1}(\lambda,a):a\in A\cap\lambda)$ and $(\{Th^{\rm loc}_{k-1}(\lambda,a):b<a<c\}:(b,c)\in A^+)$ are identical.

This theorem generalizes to $L_{\omega_1\omega}$, except that A is no longer finite; A is now an infinite tree of labeled elements with ranks B for each descending sequence B of ordinals in the quantifier rank ordinal α. It holds, too, for linear orders with unary relations. If we add unary relations to our vocabulary, there is an $\equiv^{\rm loc}_0$ class for each unary relation. With no unary relations, there is a unique $\equiv_0$ class, i.e., $(\lambda,x)\equiv_0(\lambda,y)$ for all $x\in\lambda$ and $y\in\lambda$. Patterns and words in this vocabulary are described by almost locally closed sets, in definition 3.4.2. Of course, these patterns arise even without unary predicates in the vocabulary. For instance, in the set of linear orders of the form $\lambda=\sum_{i\in\omega+Z\times\eta+\omega^*}(\eta+f(i))+\eta$ for various functions f with domain $\omega+Z\times\eta+\omega^*$ and range n, a finite number, $\equiv^{\rm loc}_4$ classes define $\{(\lambda,x):\exists i((i\in\omega+Z\times\eta+\omega^*)\wedge(x\in f(i))\wedge((f(i-1),f(i),f(i+1))=(p,q,r))\}$ for each triple (p,q,r) of numbers ≤ 9. Thus, if we choose $n=10$, then a consistent set of $\equiv^{\rm loc}_4$ sets is a set of triples (p,q,r) which can be strung together in a consistent way. There exist consistent sets of $\equiv^{\rm loc}_4$ classes for which the smallest minimal almost locally closed sets are very long.

This theorem continues the following line of research: The decidability of certain linear orders was studied in [12]. In [3] we find semimodels and the theorem that it is not easier to decide the theory of a semimodel and to relate that theory to that of a (infinite, normal, non-semi) model containing it, than

to decide the theory of linear order. The decidability of the theory of linear order was proved in [7] using Ramsey's theorem and noting the monotonicity of $\{Th_k(\{z \in \lambda : x < z < y\}) : a < x < y < b\}$ as a function of a and b. Our proof of the same theorem avoids Ramsey's theorem. When we state this theorem in its local form, it can be used to prove many sentences of $L_{\omega_1\omega}$ to be consistent:

Theorem 3.0.3: A set U of pairs of $\equiv_k$ classes is $\{(Th_k(\{a \in \lambda : a < b\}), Th_k(\{a \in \lambda : a > b\}))\}$ for some linear order λ just in case 1. there is an $\equiv_k$ class $\xi(U)$ such that for any $(\phi, \psi) \in U$, we have $\exists x(\phi^{<x} \wedge \psi^{>x}) \equiv_k \xi(U)$, and 2. there is a set W containing U and other sets of pairs of $\equiv_k$ classes such that every $V \in W$ has $\xi(V)$ as in part 1 and such that for any $V \in W$ and any element $(\phi, \psi) \in V$ there exist two elements V_0, V_1 of W such that $(\xi(V_0), \xi(V_1)) = (\phi, \psi)$ and $V_0 + \{(\emptyset, \emptyset)\} + V_1 = V$.

In [1] we find a modification of the construction in [7] to generate a family of linear orders which intersects every $\equiv_k$ class and which all have not only decidable, but finitely axiomatizable theories. Theorem 0.3 permits the elimination of Ramsey's theorem also from our construction of a finitely axiomatizable model in any $\equiv_k$ class. A more practical sort of effectiveness can also be obtained: In [9] we learn that the theory of linear order is (decidable, but) intractable. In [5] we find an enumeration of the $\equiv_3$ classes of linear orders and the statement that current methods of deciding the theory of linear order cannot enumerate the $\equiv_4$ classes of linear orders. As a proof of concept we wrote a simple computer code which enumerates the 82988077686330 $\equiv_4$ classes of linear orders. On my laptop, it took the program two minutes to write those $8E13$ sentences. There are $4E23769$ sets of pairs of $\equiv_3$ classes, almost all of which are inconsistent. The computer avoids these by eliminating small inconsistent sentences, and then ignoring any theory which contains that sentence. We cannot list the $\equiv_5$ classes of linear order, but we can describe them both locally and globally. Size, not complexity, prevents the enumeration of $\equiv_5$. From the point of view of finite model theory, the most interesting results are the implications for semimodels. semimodels are finite strings which often have interesting relationships to infinite models. semimodels admit addition and multiplication, they can code the formation of a model by iterated application of functions (e.g., Skolem functions), and they handle local information quite easily.

3.1 Decidability: the consistency game

If ϕ and ψ are $\equiv_k$ classes of linear orders, then $\{\lambda + \mu : \lambda \in \phi, \mu \in \psi\}$ is contained in a single $\equiv_k$ class, which we call the sum of ϕ and ψ. If ϕ is an $\equiv_k$ class of linear orders, choose a sentence ϕ_0 in which the variable x does not appear which defines ϕ (that is, each linear order λ models ϕ_0 just in case it is in ϕ), and let $\phi^{<x}$ be the formula, with the variable x free, obtained from ϕ_0 by replacing every subformula $\exists y(\psi)$ by $\exists y((y < x) \wedge \psi)$ and replacing every subformula $\forall y(\psi)$ by $\forall y((y < x) \rightarrow \psi)$.

We define $\phi^{>x}$ similarly. We define $\phi^{(x,y)}$ similarly – by choosing a formula ϕ_0 which defines the $\equiv_k$ class ϕ and in which neither x nor y appears, and in ϕ_0 we replace every subformula $\exists z(\psi)$ by $\exists z((x < z < y) \wedge \psi)$ and replace every subformula $\forall z(\psi)$ by $\forall z((x < z < y) \rightarrow \psi)$. Because $\lambda \models_{[a/x]} \phi^{<x}$ just in case $\{m \in \lambda : m < a\} \models \phi$, we can read $\phi^{<x}$ as "ϕ holds left of x."

For any set U of pairs of $\equiv_k$ classes, let U_0 be the set of formulas $\{\exists x(\phi^{<x} \wedge \psi^{>x}) : (\phi, \psi) \in U\}$ and let $U_1 = \{\neg\exists x(\phi^{<x} \wedge \psi^{>x}) : (\phi, \psi)$ is a pair of $\equiv_k$ classes not in $U\}$, and let $\sigma_U = \wedge\{U_0, U_1\}$. For any $\equiv_{k+1}$ class μ with linear order $\lambda \in \mu$, the set $U(\lambda) = \{$pairs of $\equiv_k$ classes $(\phi, \psi) : \lambda \models \exists x(\phi^{<x} \wedge \psi^{>x})\}$ is such that $\sigma_{U(\lambda)}$ defines μ. Now $U(\lambda)$ depends only on the $\equiv_{k+1}$ class of μ (more, $\lambda_0 \equiv_{k+1} \lambda_1$ holds just in case $U(\lambda_0) = U(\lambda_1)$) so we write $U(\mu)$. The function from μ to $U(\mu)$ and from U to σ_U are inverses, i.e., μ is defined by $\sigma_{U(\mu)}$ and $U(\sigma_V) = V$ for any consistent set V of pairs of $\equiv_k$ classes.

If W is a set of sets U of pairs of $\equiv_k$ types for which there exists an $\equiv_k$ class $\xi(U)$ such that for any $(\phi, \psi) \in U$, $\exists x(\phi^{<x} \wedge \psi^{>x}) \equiv_k \xi(U)$, then we define addition on W: $U + V =$

$$\{(\phi, \psi + Th_k(1) + \xi(V)) : (\phi, \psi) \in U\} \cup \{(\xi(U) + Th_k(1) + \phi, \psi) : (\phi, \psi) \in V\}.$$

If σ_U and σ_V are consistent sentences, and hence elements of $\equiv_{k+1}$, then we define $\sigma_U + \sigma_V$ as above to be the $\equiv_{k+1}$ class of $\lambda + \mu$ for any/all λ such that $\lambda \models \sigma_U$ and any/all μ such that $\mu \models \sigma_V$. The notion of addition we have now defined for sets of pairs of $\equiv_k$ classes then agrees with addition on $\equiv_{k+1}$ classes: $\sigma_{U+V} = \sigma_U + \sigma_V$. So addition on W extends addition of $\equiv_{k+1}$ classes to some inconsistent sentences σ_U. This is important because we will search for the consistent sentences in W, using W's structure as a monoid.

Definition 3.1.1: The consistency game is as follows: On each turn, the game state is a finite sequence $(c_i : i < n)$ of constants and a sequence $(U_i : i \leq n)$ of sets of pairs of $\equiv_k$ classes each of which has an $\equiv_k$ class $\xi(U_i)$ such that for any $(\phi, \psi) \in U_i$, $\exists x(\phi^{<x} \wedge \psi^{>x}) \equiv_k \xi(U_i)$. The first player chooses $i \leq n$ and an element of U_i. The second player then adds a new constant c left of c_i and right of c_{i-1} and chooses two sets V_0 and V_1 of pairs of $\equiv_k$ classes so that for each $i < 2$ there exists an $\equiv_k$ class $\xi(V_i)$ such that for any $(\phi, \psi) \in V_i$, $\exists x(\phi^{<x} \wedge \psi^{>x}) \equiv_k \xi(V_i)$. Player II loses unless the following conditions hold, in which case we say that player II has survived this round:

- $(\xi(V_0), \xi(V_1))$ is the element of U_i which player I chose, and
- $V_0 + \{(\emptyset, \emptyset)\} + V_1 = U_i$.

If player II has survived, then the game continues, with its game state $(c'_j : j < n + 1)$ and $(U'_j : j \leq n + 1)$ where $c'_j = c_j$ if $j < i$, $c'_{j+1} = c_j$ if $j \geq i$, and $c'_i = c$, the new constant; $U'_j = U_j$ if $j < i$, $U'_{j+1} = U_j$ if $j > i$, and $U'_i = V_0$, $U'_{i+1} = V_1$, where U_i has been replaced by V_0 and V_1. The initial state has $n = 0$, an empty sequence of constants, and a single set U_0 of pairs of $\equiv_k$ classes.

Now we prove theorem 3.0.3: If W exists as in the statement of the theorem, then player II can play the consistency game indefinitely. If player II can play the consistency game indefinitely, and if player I exhausts every set U of pairs of $\equiv_k$ classes which is ever created during the game, then the set of constants played, with the ordering on each pair c_a, c_b determined at the moment when the latter was added to the set of constants, is a linear order C; we will prove that $C \models \sigma_{U_0}$. How can player I "exhaust" the set U if, when player I plays the first element of U, that set is immediately replaced by a pair of sets, V_0, V_1?

Each element of the sum $V_0 + \{(\emptyset, \emptyset)\} + V_1$ corresponds to an element of U. In particular, $\{(\xi(V_0) + \emptyset, \emptyset + \xi(V_1))\}$ corresponds to the element that player I chose, after which U was replaced by V_0 and V_1. We say that player I exhausts U if player I plays elements in U, or in a summand such as V_0 or V_1 in a sequence of sets of pairs of $\equiv_k$ classes which sums to U, so that element of the summand corresponds to the desired element of U. Consider an interval T in C – either the interval left of c, right of c, or between c_a and c_b. Suppose that the parameters defining the interval existed already on the n-th move, as c_i, for $i < n$, or as $c_{i_0} < c_{i_1}$, for $i_0 < i_1 < n$. Define U_T to be the sum of U_i over the interval - $(\sum_{j<i} U_j + \{(\emptyset, \emptyset)\}) + U_i$ for the interval left of c_i, $(\sum_{j>i, j<n} U_j + \{(\emptyset, \emptyset)\}) + U_n$ for the interval right of c_i, or $(\sum_{j>i_0, j<i_1} U_j + \{(\emptyset, \emptyset)\}) + U_{i_1}$ for the interval between c_{i_0} and c_{i_1}. Now each constant in the interval T in C corresponds to an element of U_T, since either the constant already existed in $(c_i : i < n)$, in which case there is a summand in U_T for it, or the constant was created on the n-th move or later. In that case, when the constant is played, some U will be split into V_0 and V_1; each element of that U corresponds to some elements of U_T. On the other hand, if player I exhausts each set of pairs of $\equiv_k$ classes which is created, then U_T will be exhausted when each of its summands is exhausted. At that moment, there will be total functions mapping the interval T in C into the set U_T, and the set U_T into C, so that for $c \in T$ and $(\phi, \psi) \in U_T$, $(c, (\phi, \psi))$ being in either function or its inverse implies that the set U_T can be split into intervals V_0 and V_1 in W so that $\xi(V_0) = \phi$ and $\xi(V_1) = \psi$ and $V_0 + Th_k(1) + V_1 = U_T$. Now we prove the following: for each $j < k$, for each interval T in C, $T \equiv_j \xi(U_T)$. We prove this simultaneously for all intervals T, by induction on j – the correspondence between elements of U_T and elements of T such that (ϕ, ψ) corresponding to $c \in T$ implies that the interval of T left of c satisfies $Th_j(\phi)$ and the interval of T right of c satisfies $Th_j(\psi)$ is enough to show that T satisfies $Th_{j+1}(\xi(U_T))$. The base case, $j = 0$ is in fact the same argument: if U_T is empty, let n be the turn immediately after the last parameter was defined. Then either T is the interval left of c_0, or right of c_n, or between c_i and c_{i+1} (if the parameters are not adjacent, then U_T contains a nonempty summand $\{(\emptyset, \emptyset)\}$, and so it is not empty!). So U_T is, then $U_i = \emptyset$ for some i. Player I can never choose an element from U_i, so player II never defines a new constant that splits U_i. So the set of constants which are ever created satisfies $\sigma_{U_T} = \sigma_\emptyset = Th_{k+1}(\emptyset)$. On the other hand, if U_T is nonempty, let n be the turn immediately after the last parameter was defined. Then either there is already some constant in T, or U_T has a single summand $U_i \neq \emptyset$. Player I will play to exhaust U_i, so in particular player I will eventually play in U_i, player II will then add a constant in T. □

3.2 Local equivalence at an element

If λ and μ are linear orders and r and s are assignments of variables or constants into λ and μ, then $(\lambda, r) \equiv (\mu, s)$ just in case the domain of r and s are the same set d, and the same logical formulas, with free variables among the elements of d, are the same in both models. This holds just in case r and s induce the same ordering on d and for each $(b, c) \in d^+$, $\{a \in \lambda : b < a < c\} \equiv \{m \in \mu : b < m < c\}$. If $\equiv$ classes of theories of linear orders respect addition, then $\equiv$ classes of formulas (of finitary logic, infinitary logic, or even non-wellfounded

logic) with free variables admit addition: $\phi + \psi$ is the $\equiv$ class containing those linear orders λ and assignments s such that s assigns the free variables of ϕ and ψ into λ and for some cut $(\mu, \pi) \in \lambda^+$, s is the union of assignments t and u, where t assigns the free variables of ϕ into μ and u assigns the free variables of ψ into π, so that $\mu \models_t \phi$ and $\pi \models_u \psi$. In particular, if $\lambda \models_s \phi + \psi$, then s must assign the free variables of ϕ into λ so that they are all to the left of the free variables of ψ. We say that $\equiv$ respects addition if $\phi \equiv \phi_0$ and $\psi \equiv \psi_0$ imply that $\phi + \psi \equiv \phi_0 + \psi_0$. Finitary and infinitary quantifier-rank classes $\equiv_k$ and $\equiv_\alpha$, as well as nonwellfounded infinitary quantifier rank $\equiv_\lambda$ respect addition.

Definition 3.2.1: If $\equiv$ respects addition, we define left equivalence: $\phi \equiv^{\text{left}} \psi$ if there is some $\equiv$ class γ such that for all $\equiv$ classes α and β and all $\equiv$ variations $\alpha_0 \equiv \alpha$ and $\beta_0 \equiv \beta$,

$$\phi + \alpha + \gamma + \beta \equiv \psi + \alpha_0 + \gamma + \beta_0.$$

Likewise, $\phi \equiv^{\text{right}} \psi$ if there is some $\equiv$ class γ such that for all $\equiv$ and classes α and β and all $\equiv$ variations $\alpha_0 \equiv \alpha$ and $\beta_0 \equiv \beta$,

$$\alpha + \gamma + \beta + \phi \equiv \alpha_0 + \gamma + \beta_0 + \psi.$$

Finally, for any linear orders λ and μ and assignments r and s of a nonempty domain into λ and μ, we say $\lambda \equiv^{\text{loc}} \mu$ just in case

$$\lambda(\equiv^{\text{left}})^{\text{right}}\mu.$$

The following are properties of $\equiv^{\text{left}}$ and $\equiv^{\text{loc}}$:

- If $\equiv$ is an equivalence relation, then $\equiv^{\text{left}}$ is, too.
- If $\equiv$ respects addition, then $\equiv^{\text{left}}$ does, too.
- $(\equiv^{\text{left}})^{\text{left}}$ is the same as $\equiv^{\text{left}}$.
- $(\equiv^{\text{left}})^{\text{right}}$ is the same as $(\equiv^{\text{right}})^{\text{left}}$.
- If ϕ and ψ have at least one free variable, then $\phi \equiv^{\text{left}} \psi$ and $\phi \equiv^{\text{right}} \psi$ together imply $\phi \equiv \psi$.
- If ϕ and ψ have at least one free variable, then for $(\lambda, a_i)_{i<n} \in \phi$ and $(\mu, m_i)_{i<n} \in \psi$, $(\lambda, a_0, \ldots a_{n-1}) \equiv^{\text{loc}} (\mu, m_0, \ldots m_{n-1})$ holds just in case:
  - $\{a \in \lambda : a < a_0\} \equiv^{\text{right}} \{m \in \mu : m < m_0\}$,
  - for each $i < n-1$, $\{a \in \lambda : a_i < a < a_{i+1}\} \equiv \{m \in \mu : m_i < m < m_{i+1}\}$,
  - $\{a \in \lambda : a > a_{n-1}\} \equiv^{\text{left}} \{m \in \mu : m > m_{n-1}\}$.
- $\equiv_0^{\text{loc}}$ and $\equiv_1^{\text{loc}}$ are trivial $\equiv$ relations, i.e., for $i < 2$, $(\lambda, r) \equiv_i^{\text{loc}} (\mu, s)$ holds just in case r and s have the same domain and induce the same ordering on it.

- If $\equiv$ is an equivalence relation on theories of linear order which respects addition and if there is an $\equiv$ class σ_0 such that for every other $\equiv$ class δ it holds that $\sigma_0 + \delta + \sigma_0 \equiv \sigma_0$, then for any $\equiv^{\text{left}}$ class ϕ, the sum $\phi + \sigma_0$ is *inextensible* in the sense that for any $\equiv$ class ψ, $\phi + \sigma_0 + \psi \equiv^{\text{left}} \phi + \sigma_0$.

- if σ_0 exists as in the previous item, then for any $\equiv$ classes ϕ and ψ with no free variables, $\phi(\equiv^{\text{left}})^{\text{right}}\psi$.

- Give $\equiv_\lambda$ some linear ordering E. On a dense linear ordering η_0 such that every interval has cardinality $> |E|^{|\lambda|}$, we can define a function to E so that the sum $\sigma_0 = \sum_{a \in \eta_0} f(a)$; if $f^{-1}(a)$ is dense, then σ_0 satisfies $\sigma_0 + \delta + \sigma_0 \equiv_\lambda \sigma_0$.

- If σ_0 exists as in the previous items, then $\sigma_0 \times \omega$ or indeed any linear order γ such that $\sigma_0 + \gamma = \gamma$ is sufficient to prove $\equiv^{\text{left}}$ in the following theorem: any $\equiv^{\text{left}}$ class is $\equiv^{\text{left}} \vee U$ for U a set of inextensible $\equiv^{\text{left}}$ classes (adding any $\equiv$ class to the right leaves each of these $\equiv^{\text{left}}$ classes unchanged).

- There are three $\equiv_2^{\text{left}}$ classes of formulas with a single free variable: those (λ, a) such that a has an immediate successor, those (λ, a) such that a is the limit of a sequence descending from above, and those (λ, a) such that a is the greatest element of λ. The third $\equiv_2^{\text{left}}$ class is the disjunction of the first two, which are inextensible.

3.3 Labeling firsts and lasts

For any linear order λ, let $\lambda^+ = \{(b, c) : b \cup c \subseteq \lambda$ and $\forall d \in b \forall e \in c(d < e)$ and $\forall a \in \lambda((\exists d \in b(a \leq b)) \vee (\exists d \in c(c \leq a)))\}$ modulo the equivalence $(b, c) \equiv (b', c')$ which holds just in case $\forall d \in b(\exists e \in b'(d \leq e))$ and $\forall d \in b'(\exists e \in b(d \leq e))$ be the set of *cuts* in λ. There is a natural ordering on $\lambda \cup \lambda^+$ induced from the ordering on λ: $a < (b, c)$ holds just in case $\exists d \in b(a \leq d)$; $(b, c) < (d, e)$ holds just in case $\exists f \in b(\forall g \in d(g < f))$.

If the range of a function is a linear order then that order is induced on the function itself. If b is a function with range $\subseteq \lambda \cup \lambda^+$ then let $\sup b$ be the cut $(\{a \in \lambda : \exists c \in b(a \leq c)\}, \{a \in \lambda : \forall c \in b(c < a)\})$. We define the *infimum* similarly. If b and c are functions into $\lambda \cup \lambda^+$, let $\lambda^{(b,c)} = \{a \in \lambda : \sup b < a < \inf c\}$.

Definition 3.3.1: If I is a set of labels with any ordering and (b, c) is any cut $(b, c) \in I^+$, then for each formula τ with a single free variable (rather, for every $\equiv^{\text{loc}}$ class τ which is inextensible in the universe of $\equiv$ classes of intervals of λ) we create four new *labels*, the elements: the least τ in (b, c) and the greatest τ in (b, c) and the cuts: the unrealized infimum of an unbounded descending sequence of τ in (b, c) and the unrealized supremum of an unbounded ascending sequence of τ in (b, c). We will abbreviate these phrases with the symbols: $l\tau \in (b, c)$, $g\tau \in (b, c)$, $d\tau \in (b, c)$, and $a\tau \in (b, c)$. When the context (b, c) can be inferred, we will write them simply as $l\tau, g\tau, d\tau, a\tau$ All *labels* are constructed in a wellfounded way by this rule, from $f = \emptyset$.

For instance, if E_0 and E_1 and E_2 and E_3 are two equivalence relations on theories of linear order, and if for each $i < 4$, τ_i is an equivalence class, then

$l\tau_0 \in (b, \emptyset)$, where b is a function with domain $\{l\tau_1 \in (g\tau_2 \in (\emptyset, \emptyset), g\tau_3 \in (\emptyset, \emptyset))\}$ is a label.

The element $l\tau_0 \in (b, c)$ is the least element of type τ_0 above $\sup b$... there is no need to refer to c, unless we want to say that the least τ_0 above $\sup b$ happens to be below $\inf c$. Viewing $l\tau_0 \in (b, c)$ as the least (or infimum of a descending sequence of) τ_0 above (say) the least τ_1 above (say) the greatest (or supremum of an ascending sequence of) τ_2 below ..., we can write the set of all labels as a tree. The first rank of the tree contains the least (or infimum) and greatest (or supremum) appearance of each type τ; the next rank of the tree contains the least appearance of each type above an element of the first rank, or the greatest appearance of each type below an element of the first rank. From the tree structure and the ordering induced by λ on its branches, we can write each label in the form given in the preceding definition.

Definition 3.3.2: If λ is a linear order and $\equiv$ is an equivalence relation on theories of linear order and I is an assignment of labels into $\lambda \cup \lambda^+$ then the $\equiv^{\text{loc}}$-refinement of I is the smallest assignment containing I and for each $(b, c) \in I^+$, one or two elements or cuts to indicate the definable least (or infimum) and/or greatest (or supremum) of the elements of $\equiv^{\text{loc}}$ class τ in (b, c). If τ is realized in (b, c) then τ's least element(s) are definable just in case one of the following holds:

- $b = \emptyset$, or if that fails, then
- there is a maximal element of b of any form except $a\tau' \in (e, f)$ for τ' a *high-order* equivalence class – i.e., an equivalence class in an equivalence relation which is equal to, or refines, $\equiv$, or that fails and
- elements of type τ are bounded in $a\tau' \in (e, f)$ below some element of λ.

If τ's least element(s) are definable, then I' assigns either $l\tau \in (b, c)$ or $d\tau \in (b, c)$ into $\lambda^{(b,c)}$:

- the label $l\tau \in (b, c)$ assigned to the least $h \in \lambda^{(b,c)}$ such that $(\lambda, h) \in \tau$ if there is a least such, or
- the label $d\tau \in (b, c)$ assigned to the greatest cut (g, h) such that h contains all elements of $\lambda^{(b,c)}$ of $\equiv^{\text{loc}}$ class τ, if that set h has no least element.

Similarly, for all τ realized in (b, c), the greatest element(s) of $\equiv^{\text{loc}}$ class τ are definable just in case:

- $c = \emptyset$, or $c \neq \emptyset$ and
- c has a minimal label, and that label is not $d\tau' \in (e, f)$ for τ' a high-order equivalence class, or c has either no least element or has the tricky least element just described, but
- elements of type τ are bounded in $d\tau' \in (e, f)$ above some element of λ.

If the greatest element(s) of $\equiv^{\text{loc}}$ class τ are definable, then I' contains either:

- the label $g\tau \in (b, c)$ assigned to the greatest $h \in \lambda^{(b,c)}$ such that $(\lambda, h) \in \tau$, or

- the label $a\tau \in (b,c)$ assigned to the least cut (g,h) such that g contains all elements of $\lambda^{(b,c)}$ of $\equiv^{\mathrm{loc}}$ class τ, if that set g has no greatest element.

It is sufficient to consider only inextensible $\equiv^{\mathrm{loc}}$ classes τ, since if $\tau = \vee U$, then if the least element of type τ is definable, then so are all the elements of U, and $\inf \tau$ is $\inf_{\tau' \in U} \inf \tau'$.

For instance, if we write e for the unique $\equiv_0^{loc}$ class, the seven $\equiv_2$ classes of linear order can be enumerated as:

$$le < ae, de < ge, de < ae, \emptyset, le = ge, le < \emptyset < ge, le < \exists x e(x) < ge.$$

That this list is a complete list of $\equiv_2$ classes of linear orders will be proved later by appeal to theorem 3.0.3: a single set W contains five of the seven consistent sets of pairs of $\equiv_2$ classes, another set W contains three; the remaining nine sets of pairs of $\equiv_1$ classes are quickly proved inconsistent. For quantifier rank $k > 2$, however, theorem 3.0.3 is quaint and useless; we enumerate $\equiv_k$ classes by describing trees of labels and $\equiv_{k-1}^{\mathrm{loc}}$ classes.

Definition 3.3.3: Order the finite sets σ of natural numbers lexicographically: by the largest element, then the next largest, etc. Let $I_\emptyset(\lambda) = \emptyset$. For each finite set σ of natural numbers, let σ' be its immediate successor in the lexicographical order[1] and n be the least element of σ' and let $I_{\sigma'}(\lambda)$ be the $\equiv_n^{\mathrm{loc}}$-refinement of I.

Consider the EF game of length k between two linear orders μ_0 and μ_1 which realize the same $\equiv_{k-1}^{\mathrm{loc}}$ classes. The following lemma explains how player I can use the tree of labels to find non-local differences between μ_0 and μ_1. Later, we will prove that this strategy is complete – $\equiv_k$ holds if player I cannot find a way to use this lemma; $\not\equiv_k$ holds if player I can.

Lemma 3.3.1: Player I has a winning strategy in the game $EF_k(\mu_0, \mu_1)$ game if after $a_i \in \mu_i$ are chosen on the first move, the condition $(1 \wedge 2 \wedge 3) \vee (4 \wedge 5 \wedge 6)$ holds at some rank $\sigma \subseteq k-1$, $\sigma \neq k-1$, in the tree of labels:

1. $k-2 \notin \sigma$ and
2. $I_\sigma(\mu_0)$ and $I_\sigma(\mu_1)$ induce the same order on the same tree of labels and a_0 and a_1 are in the same cut (b,c) in $(I_\sigma(\mu_i))^+$, and
3. for some $i < 2$, some $\equiv_{k-2}^{\mathrm{loc}}$ class ρ is realized in μ_i between $\sup b$ and a_i and is not realized in μ_{1-i} between $\sup b$ and a_{1-i} and ρ is definable above $\sup b$ in the sense of definition 3.3.2, or
4. $k-2 \in \sigma$ and
5. $I_{\{i:i<k-2\}}(\mu_0)$ and $I_{\{i:i<k-2\}}(\mu_1)$ induce the same order on the same tree of labels and a_0 and a_1 are in the same cut (b_0, c_0) in $(I_{\{i:i<k-2\}}(\mu_i))^+$, and

[1] The sets $\sigma < \sigma'$ are a pair of immediate predecessor and successor just in case $\sum_{i\in\sigma'} 2^i = 1 + \sum_{i\in\sigma} 2^i$.

6. For some $\equiv^{\mathrm{loc}}_{k-2}$ class ρ and some $i < 2$, there is some $p_i \in \mu_i^{(b,a_i)}$ such that for all $p_{1-i} \in \mu_{1-i}^{(b,a_{1-i})}$, if $(\mu_0, p_0) \in \rho$ and $(\mu_1, p_1) \in \rho$ and p_0 and p_1 are in (b_0, c_0) and if $I_{\sigma\setminus\{k-2\}}(\mu_i^{>p_i})$ and $I_{\sigma\setminus\{k-2\}}(\mu_i^{>p_{1-i}})$ assign the same labels in the same order, then conditions $(1 \wedge 2 \wedge 3) \vee (4 \wedge 5 \wedge 6)$ hold at rank $\sigma \setminus \{k-2\}$ in the tree of labels after $a_0 \in \mu_0^{>p_0}$ and $a_1 \in \mu_1^{>p_1}$ are played on the first move in the game $EF_{k-1}(\mu_0^{>p_0}, \mu_1^{>p_1})$.

Proof: Suppose condition $(1 \wedge 2 \wedge 3)$ holds. Player I plays the element of type ρ in μ_i between $\sup b$ and a_i. Player II must respond with an element in $\equiv^{\mathrm{loc}}_{k-2}$ class ρ, since $k-2$-many moves will remain after this second move in $EF_k(\mu_0, \mu_1)$. By condition 3, player II will only find such an element below $\sup b$ in μ_{1-i}. If this were a winning second move for player II in $EF_k(\mu_0, \mu_1)$, then it is a winning first move in $EF_{k-1}(\mu_0^{<a_0}, \mu_1^{<a_1})$. But by theorem 3.3.1 for $k-1$ in place of k, the first move of player II in $EF_{k-1}(\mu_0^{<a_0}, \mu_1^{<a_1})$ must be in the same interval in $I_\sigma(\mu_i^{<a_i})$; since $k-2 \notin \sigma$, $b \subseteq I_\sigma(\mu_i^{<a_i})$. Suppose, on the other hand, that conditions $(4 \wedge 5 \wedge 6)$ hold. Player I then plays the element p_i mentioned in condition 6. Player II must answer with an element p_{1-i} of the same $\equiv^{\mathrm{loc}}_{k-2}$ class and (by theorem 3.3.1 for $k = 1$ in place of k) in the same interval of $I_{\{i:i<k-2\}}(\mu_i)$. If now the tree of labels $I_{\sigma\setminus\{k-2\}}(\mu_i^{>p_i})$ differ, then player II has lost; if they agree but the elements a_i are in different cuts $(b, c) \in (I_{\sigma\setminus\{k-2\}}(\mu_i^{>p_i}))^+$, then by theorem 3.3.1, player II has lost. Finally, if these data are the same, then condition 6 implies that we can now re-apply the lemma to rank $\sigma \setminus \{k-2\}$ of the tree of labels $I_{\sigma\setminus\{k-2\}}(\mu_i^{>p_i})$. But since $\sigma \neq k-1$, eventually it will be conditions $1 \wedge 2 \wedge 3$ which hold, rather than conditions $4 \wedge 5 \wedge 6$. □

If after the first move of the EF game identifies $a_i \in \mu_i$, player I finds that the lemma holds for σ and $n < k-1$ such that $n \notin \sigma$, a_i are in the same cut (b, c) of $I_{\sigma\setminus n}(\mu_i)$, and there is an $\equiv^{\mathrm{loc}}_n$ class ρ which is definable above $\sup b$ so that an element of $\equiv^{\mathrm{loc}}_n$ class ρ exists between $\sup b$ and a_i but not between $\sup b$ and a_{1-i} then we call ρ the *anomaly* between $\sup b$ and a_i. The following theorem's modest claim: "if player II has a winning strategy, and player II disrespects the next refinement, this leaves a game in which player I has a winning strategy" can be repeated over any series of $\equiv_n$ refinements, when n is not in the index set σ, producing an index set with n as its least element. This can produce trees with various ranks, but the one way to produce a maximal tree is to consider each $\sigma \subseteq k-1$ in lexicographical order. This then implies that if player II has a winning strategy, player II must respect $I_{\{i:i<k-1\}}$, so that $\mu_0 \equiv_k \mu_1$ implies that the same labels are sent into μ_0 and μ_1 in the same order.

Theorem 3.3.1: If player II has a winning strategy in $EF_k(\mu_0, \mu_1)$, then for each $\sigma \subseteq k-1$,

- $I_\sigma(\mu_0)$ and $I_\sigma(\mu_1)$ induce the same order on the same tree of labels, and
- if player I plays the first move at the image of a label in one model, then either player II plays the image of that label in the other model or player I has a winning strategy in the remainder of the game, and
- if $(b, c) \in (I_\sigma(\mu_0))^+$ and player I plays the first move in $\mu_i^{(b,c)}$, then either player II plays in $\mu_{1-i}^{(b,c)}$ or player I has a winning strategy in the remainder of the game.

Proof by induction on $\sigma \subseteq k-1$, ordered lexicographically: If $\sigma = \emptyset$, then the first two conditions require nothing and the third condition, with $(b,c) = (\emptyset,\emptyset)$ and $\mu_i^{(b,c)} = \mu_i$, is nothing more than one of the rules of the EF game: if player I plays in μ_i then player II answers in μ_{1-i} or loses. Now we suppose the theorem is proved up to σ_0, the immediate predecessor of σ in the lexicographical ordering, and we prove the theorem for σ itself. Since $I_{\sigma_0}(\mu_i)$ must be respected by player II on the first move, we can define a cut $(b,c) \in (I_{\sigma_0}(\mu_i))^+$ so that the first move is played between $\sup b$ and $\inf c$. To apply lemma 3.3.1, note that the least element of σ is not in σ_0. The least element of σ is that n for which respecting I_σ means respecting I_{σ_0} and respecting the first and last occurrences of elements of each definable $\equiv_n^{\mathrm{loc}}$ class. Now n is the *least* number not in σ_0 – $1 + \sum_{i<n} 2^i = 2^n$ – but the lemma applies to the *greatest* number not in σ_0. But if there is an anomalous $\equiv_n^{\mathrm{loc}}$ class ρ between $\sup b$ and a_i, the first played element, then there is an anomalous $\equiv_{n+1}^{\mathrm{loc}}$ class, since the $\equiv_{n+1}^{\mathrm{loc}}$ class of any element realizing ρ isn't realized in μ_{1-i} between $\sup b$ and a_{1-i}, and likewise there is an anomalous $\equiv_m^{\mathrm{loc}}$ class, for every $m > n$.

Player I's goal is to preserve a *winning condition*: that the first moves $a_i \in \mu_i$ were played in the same cut $(b,c) \in (I_{\sigma_0}(\mu_i))^+$ and an anomaly exists - i.e., some $\equiv_m^{\mathrm{loc}}$ class ρ is realized in μ_i between $\sup b$ and a_i and not realized in μ_{1-i} between $\sup b$ and a_{1-i}, for $n \notin \sigma_0$. Player I then plays $p_i \in I_{\{k-2\}}(\mu_i) \setminus I_{\{i:i<k-2\}}(\mu_i)$; player II must preserve $I_{\sigma_0 \setminus \{k-2\}}(\mu_i)$ above p_i, and player I finds that the anomaly has been preserved. Player I repeats this until there are $m+1$-many moves left. That is, player I plays the first move to the lower end of the interval in $I_{\{k-2\}}$ which contains a_i, then the lower end of the interval in $I_{\{k-2,k-3\}}$ which contains a_i, and so on, until on the j-th ($j = k-1-m$) turn player I plays the lower end of the interval (b_j, c_j) in $I_{\{k-2,k-3,\ldots,k-j\}}$ for $j \leq k-1-m$ in which a_i occurs. We will discuss 1. how player I can play close enough to a cut and below it (or above it) to define all the $\equiv_*^{\mathrm{loc}}$ classes which are definable above it (or below it) and 2. which model player I should play in so as to prevent new $\equiv_m^{\mathrm{loc}}$ classes from entering the interval between $\sup b$ and a_{1-i}. We will assume, throughout, that player II plays, on the j-th move, an element of the same $\equiv_j^{\mathrm{loc}}$ class as player I. We will prove that player I can preserve a winning condition – the existence of the same anomaly – until there are $m+1$-many moves left.

Player I's winning strategy after playing at the anomaly: On the $k-m$-th move (after which there will be m-many moves remaining), player I will play the anomaly – an element of type ρ which exists between $\sup b$ and a_i, such that ρ is not realized in μ_{1-i} between $\sup b$ and a_{1-i}. Since there will remain m-many moves, player II must respond with an element of $\equiv_m^{\mathrm{loc}}$ class ρ. This can only be found below $\sup b$ in μ_{1-i}. If player II has played the second through $k-m-1$-th moves according to a winning strategy in the EF game of length k, the first move of which identifies $a_i \in \mu_i$ and the $k-m-1$-th move of which identifies $p_i \in \mu_i$, then $(\mu_0^{>p_0}, a_0) \equiv_{m+1} (\mu_1^{>p_1}, a_1)$. Iterating theorem 3.3.1, with k replaced by $m+1$, for all subsets of $m+1$, we find: $I_{\{j:j<m\}}(\mu_0^{>p_0})$ and $I_{\{j:j<m\}}(\mu_1^{>p_1})$ induce the same order on the same tree of labels and the first move, in which player I plays at the anomaly and player II plays below $\sup b \in \mu_i$, must respect $I_{\{j:j<m\}}(\mu_i^{>p_i})$. However, the labels of b which depend, in the tree of labels, on b_{k-1-m} or c_{k-1-m}, the lower and upper ends of the interval (b_{k-1-m}, c_{k-1-m}) can be redefined in terms of p_i – which was played near b_{k-1-m} with this goal

in mind, or in terms of a_i, which is certainly $< c_{k-1-m}$. Thus, every label in b of tree-rank $> \sigma_0$ corresponds to a label of $I_{\{j:j<m\}}(\mu_i^{(p_i,a_i)})$, where the latter labels are mapped monotonically into λ, so that no element of $\equiv_m^{\mathrm{loc}}$ class ρ exists between $\sup b$ and a_{1-i}, but yet player II must respect $\sup b$. So player II loses.

Player I's algorithm for determining which linear order to play in: If on the j-th move (for $j \geq 1$ and $j < k-m-1$) player I plans to play at $\sup b_j$, where (b_j, c_j) is the interval in $I_{\{k-2,k-3,\ldots,k-j\}}$ in which a_i occurs, player I must find an element close to $\sup b_j$ (see the next paragraph) in μ_i or μ_{1-i}, choosing the correct model so as to prevent player II from "widening" the interval between $\sup b$ and a_{1-i} to allow an element of type ρ to be realized there, eliminating the anomaly, or "narrowing" the interval between $\sup b$ and a_i to remove all elements of $\equiv_m^{\mathrm{loc}}$ class ρ there.

- Player I plays $d\tau \in (b,c)$ or $l\tau \in (b,c)$ in μ_i. Player II must respond with an element of type τ, which is $\geq$ the least element of type τ in $\mu_{1-i}^{(b,c)}$, which will preserve the defined elements of b, or will shift all defined elements monotonically to the right, and so preserve or narrow the interval between $\sup b$ and and a_{1-i}.
- Player I plays $a\tau \in (b,c)$ or $g\tau \in (b,c)$ in μ_{1-i}. Player II either plays the image of that label in μ_i, or plays some other, lower, realization of τ in $\mu_{1-i}^{(b,c)}$, shifting all the defined elements (in particular, all of b) monotonically to the left in μ_i. This preserves or widens the interval between $\sup b$ and a_i and preserves the anomaly.

Players can play close enough to any cut: To play the label $l\tau \in (b,c)$ or $g\tau \in (b,c)$ player I plays the image of that label. To play near $d\tau \in (b,c)$ or $a\tau \in (b,c)$ player I plays plays an element of type τ above $d\tau \in (b,c)$ and below $a\tau \in (b,c)$ and near the cut – closer to the cut than any other label which is not assigned to the same cut, and closer to the cut than any upper bound or lower bound which in definition 3.3.2 triggers the third condition and allows some type τ' to be definable. If player I plays x_j above the lower bound on elements of type τ' below $a\tau'' \in (b'', c'')$, then the least element of type τ' above $a\tau'' \in (b'', c'')$ is the least element of type τ' above x_j. Thus, the tree of definable labels which depend on $a\tau'' \in (b'', c'')$ for their definition and are $> a\tau'' \in (b'', c'')$ is unchanged if we replace $a\tau'' \in (b'', c'')$ by x_0.

If player II does not respect I_σ, then the winning condition is established: We examine all cases and show that player II must play above some element of each $\equiv_n^{\mathrm{loc}}$ class τ such that player I played above some element of $\equiv_n^{\mathrm{loc}}$ class τ. On the other hand, player II must play below some element of each $\equiv_n^{\mathrm{loc}}$ class τ which player I played below. This will imply that player II must respect the $\equiv_n^{\mathrm{loc}}$-refinement of I_{σ_0} or lose. Thus, if the $\equiv_n^{\mathrm{loc}}$-refinement of $I_{\sigma_0}(\mu_0)$ and the $\equiv_n^{\mathrm{loc}}$-refinement of $I_{\sigma_0}(\mu_1)$ are not identical, player II will have lost. In particular, we now prove that in each interval of $I_{\sigma_0}(\mu_i)$, the same $\equiv_m^{\mathrm{loc}}$ classes begin and end in the same order for $i = 0, 1$: First we prove that the same $\equiv_m^{\mathrm{loc}}$ classes τ are realized in any cut (b,c) in $(I_{\sigma_0}(\mu_i))^+$ and then that different $\equiv_m^{\mathrm{loc}}$ classes τ begin and end with 1. a terminal element in both linear orders or 2. no terminal element in both linear orders, and that labels for those terminal elements (or for those suprema/infima) have the same order. For if τ is realized in (b,c) in $(I_{\sigma_0}(\mu_i))^+$ and not in $\mu_{1-i}^{(b,c)}$, then player I can play τ on the first turn,

player II will play below $\sup b$ (or above $\inf c$) and player I can then exhaust $\sup b$ by playing the greatest element of $b \cap I_{\{k-2,k-3,\ldots,k-j\}}$ on the j-th move. If τ is realized a least time in $\mu_{1-i}^{(b,c)}$ but no least time in $\mu_i^{(b,c)}$, then player I plays the least element of type τ in $\mu_{1-i}^{(b,c)}$. Whatever element a_i player II plays, there will be an anomalous element of type τ in $\mu_i^{(b,c)}$ below a_i. Now we show that these firsts and lasts have the same order in μ_i and μ_{1-i}.

Suppose the $\equiv_n^{\mathrm{loc}}$-refinement of $I_{\sigma_0}(\mu_i)$ maps a label to s_i, the greatest realizations of some type, or maps a label for an unrealized supremum of an ascending sequence of realizations of that type to the cut $s_i \in \mu_i$. Suppose the $\equiv_n^{\mathrm{loc}}$-refinement of $I_{\sigma_0}(\mu_i)$ maps a label for the least realization of some type, or for an unrealized infimum of a descending sequence of realizations of that type to the cuts $r_0 \in (\mu_0)^+$ and $r_1 \in (\mu_1)^+$. We now list cases to show that the following occur in both models simultaneously: $r < s, r > s, r = s$.

- If $r < s$ holds in μ_i but not in μ_{1-i}, then player I plays the first move at a_i between r and s. By assumption, player II must respect $I_{\sigma_0}(\mu_i)$ and reply with a move $a_{1-i} \in \mu_{1-i}^{(b,c)}$ in the same cut $(b,c) \in (I_{\sigma_0}(\mu_i))^+$ for which $a_i \in \mu_i^{(b,c)}$. The type that r labels is realized in (b,c) to the left of a_i and the type that s labels is realized in (b,c) to the right of a_i. Respecting both of these conditions means playing the first move between the cuts r and s in $\mu_{1-i}^{(b,c)}$. Of course, this is only possible if those cuts have the same ordering, $r_{1-i} < s_{1-i}$.

- Suppose $s < r$ holds in μ_{1-i} and $s \geq r$ holds in μ_i. Player I plays a_{1-i} between s and r in μ_{1-i}. Whatever element a_i player II plays in $\mu_i^{(b,c)}$, either the type that r labels is realized in (b,c) to the left of a_i or the type that s labels is realized in (b,c) to the right of a_i, which is a winning condition for player I.

- Suppose $r = s$ holds in μ_{1-i} and fails in μ_i. Then $r < s$ or $r > s$ holds in μ_i. Proceed as in the previous two cases.

There remains one more case to check: suppose r and s are both the suprema of ascending sequences of different type in (b,c) in μ_i. Then $r < s$ holds just in case player I can play an element of the type which s labels, and bound all elements of type r to one side; $r = s$ holds just in case $r \not< s$ and $s \not< r$, since then they are equivalent cuts. □

This theorem proves that player II must respect $I_{(\sigma_0 \cup \{n\} \setminus n)}$ if player II must respect I_{σ_0} and $n \notin \sigma_0$ and $n < k-1$. This explains why $\equiv_k$ is equivalent to local equivalence in the intervals defined by exactly $2^{k-1}-1$ -many iterations of the refinement process. For example, if $\equiv_5$ requires player II to respect $I_{\{1,0\}}$ then we can prove that player II further respects the $\equiv_n^{\mathrm{loc}}$-refinement of this for $n = 2$ or $n = 3$. If we take the $\equiv_3^{\mathrm{loc}}$-refinement, and then the $\equiv_0^{\mathrm{loc}}$, $\equiv_1^{\mathrm{loc}}$, $\equiv_0^{\mathrm{loc}}$, $\equiv_2^{\mathrm{loc}}$, $\equiv_0^{\mathrm{loc}}$,$\equiv_1^{\mathrm{loc}}$, and $\equiv_0^{\mathrm{loc}}$-refinements, we will have a tree of labels that player II must respect. But, in fact, a much larger tree of labels must be respected, and we can build it by taking the $\equiv_2^{\mathrm{loc}}$-refinement first (and then the $\equiv_0^{\mathrm{loc}}$, $\equiv_1^{\mathrm{loc}}$, and $\equiv_0^{\mathrm{loc}}$-refinements) before taking the $\equiv_3^{\mathrm{loc}}$-refinement. It is interesting to see that the range of the smaller tree of labels is contained strictly in the range of the larger tree of labels, often with very different labels for the same element.

The largest possible tree of labels which adds a singleton $\{n\}$ and removes the set n whenever $n \notin \sigma$, for σ indexing a rank in the tree, is clearly the tree given in definition 3.3.3. However, when we argue that a certain label can't be realized, we don't need the largest possible tree of labels, but only enough labels to define the one we want to discuss. For instance, we can describe the greatest limit point below x without calling it "the greatest limit point below the immediate predecessor of x."

Lemma 3.3.2: For any linear order λ and finite set of natural numbers $\sigma \subseteq k-1$, let $I_\sigma^{\text{left}}(\lambda)$ be the intersection of $I_\sigma(\mu)$ as μ ranges over the $\equiv_k^{\text{left}}$ class of λ and let $I_\sigma^{\text{right}}(\lambda)$ be the intersection of $I_\sigma(\mu)$ as μ ranges over the $\equiv_k^{\text{right}}$ class of λ. Then

$$I_\sigma(\lambda) = I_\sigma^{\text{left}}(\lambda) \cup I_\sigma^{\text{right}}(\lambda).$$

Proof: every label in the tree of labels depends ultimately on a rank-$\{0\}$ label which is one of $l\tau > \emptyset$ or $d\tau > \emptyset$ where $\emptyset$ refers to the left end, or $g\tau < \emptyset$ or $a\tau < \emptyset$, where $\emptyset$ refers to the right end. The first two are in $I_\sigma^{\text{left}}(\lambda)$ and the latter two are in $I_\sigma^{\text{right}}(\lambda)$. Similarly, all labels which depend on labels which depend ultimately on the left end (and in the next higher rank, on $l\tau > \emptyset$ or $d\tau > \emptyset$) are in $I_\sigma^{\text{left}}(\lambda)$ and all labels which depend on labels which depend ultimately on the right end are in $I_\sigma^{\text{right}}(\lambda)$. Of course, adding $\equiv_k$ classes to the right can extend the $\equiv_n^{\text{loc}}$ classes of the labels in $I_\sigma^{\text{left}}(\lambda)$ and thereby superficially change the definition of the label, but it cannot change where each of these labels is assigned in λ. □

Theorem 3.3.2: For any linear order λ and any element $a \in \lambda$, from

- the ordered set A of $I_{\{i:i<k-1\}}(\lambda)$, the functions $(Th_{k-1}^{\text{loc}}(\lambda, a) : a \in A \cap \lambda)$ and $(\{Th_{k-1}^{\text{loc}}(\lambda, a) : b < a < c\} : (b, c) \in A^+)$, and
- the location of a in A, and the $\equiv_{k-1}^{\text{loc}}$ class of (λ, a),

we can construct the structures in the first item, with $k-1$ replacing k, and either $\{d \in \lambda : d < a\}$ or $\{d \in \lambda : d > a\}$ replacing λ.

Proof: The left labels $I_{\{i:i<k-2\}}^{\text{left}}(\{d \in \lambda : d < a\})$, those labels which depend for their definition on the left end of the interval $\{d \in \lambda : d < a\}$, correspond to labels of relatively extended $\equiv_n^{\text{loc}}$ classes in $I_{\{i:i<k-1\}}^{\text{left}}(\lambda)$. We can compute the $\equiv_{k-2}^{\text{loc}}$ class of each of those left labels as the truncation of Th_{k-2} of the $\equiv_{k-1}^{\text{loc}}$ class given in the first item, above, truncated at a. That truncation can be performed by altering $I_{\{i:i<k-1\}}^{\text{left}}(\lambda)$ through every possible $I_{\{i:i<k-1\}}^{\text{left}}(\lambda')$ which agrees with $I_{\{i:i<k-1\}}^{\text{left}}(\lambda)$ to the left of a, and taking $\vee$ of the $\equiv_{k-2}^{\text{loc}}$ class in each model, or by treating the $\equiv_{k-2}^{\text{loc}}$ class as a formula, and removing that part of it which is satisfied to the right of a. In this way, we replace the $\equiv_{k-2}^{\text{loc}}$ class (λ, a) with the $\equiv_{k-2}^{\text{loc}}$ class of $(\{d \in \lambda : d < a\}, b)$. The right labels $I_{\{i:i<k-2\}}^{\text{right}}(\{d \in \lambda : d < a\})$ can be read from the $\equiv_{k-1}^{\text{loc}}$ class of (λ, a). It remains to determine an ordering of the union $I = I_{\{i:i<k-2\}}^{\text{left}}(\{d \in \lambda : d < a\})$ and $I_{\{i:i<k-2\}}^{\text{right}}(\{d \in \lambda : d < a\})$, and to determine which $\equiv_{k-2}^{\text{loc}}$ classes exist in each cut in I^+.

For any cut $(b_0, b_1) \in (I^{\text{left}}_{\{i:i<k-2\}}(\{d \in \lambda : d < a\}))^+$ and for any cut $(c_0, c_1) \in (I^{\text{right}}_{\{i:i<k-2\}}(\{d \in \lambda : d < a\}))^+$, we compute from (c_0, c_1) the set C_1 of sequences $(\tau_\sigma : \sigma \subseteq k-2)$ for which there exist elements of type τ_σ in order between (c_0, c_1) and a and the set C_0 of sequences which do not exist between (c_0, c_1) and a. We reverse these: There is an element of $\equiv^{\text{loc}}_{k-2}$ class τ below C_1 below a, but not below C_0 below a just in case for each sequence $(\tau_\sigma : \sigma \subseteq k-2)$ of $\equiv^{\text{loc}}_n$ classes, where n is the least element of $k-2$, we form the following label and find it to be $< a$:

- the least element(s) of $\equiv^{\text{loc}}_{k-2}$ class τ above b_0
- for each $\equiv^{\text{loc}}_n$ class, in descending lexicographical order, from $\tau_{\{j:j<k-1\}}$ to $\tau_\emptyset$, the least element of type τ above the previous label.

Similarly, we invert the dependency of $\exists \tau \in (c_0 \in \ldots g\tau_0 \in (\emptyset, a), c_1 \ldots a\tau_0 \in (\emptyset, a))$ on a into a dependency of a on sequences in $I^{\text{left}}_{\{i:i<k-2\}}(\{d \in \lambda : d < a\})$. For instance, if a is very far from the left end, then $I^{\text{left}}_{\{i:i<k-2\}}(\{d \in \lambda : d < a\}) < I^{\text{right}}_{\{i:i<k-2\}}(\{d \in \lambda : d < a\})$; the total order on their union is that the one linear order simply precedes the other.

On the other hand, for each $\equiv^{\text{loc}}_{k-2}$ class τ there is a sequence of labels inverting $c \in c_0 \cup c_1$ such that there is an element of type τ above b_0 and below b_1 and above c in $\{d \in \lambda : d < a\}$ just in case the location of a in $I_{\{}n : n < k-1\}$ shows $a <$ the label. Let c' be the label on which the assignment of c depends: c is the least τ_0 above c' or the greatest τ_0 below c'. If c is in c_0, then we define the next label to be the greatest τ_0 below the previous label in the sequence. If c is in c_1, then we define the least τ_0 above the previous label. Repeat this for each label in the sequence of dependency, until c is the greatest τ_i below a (i.e., there are no more labels on which c depends). Define the next label to be the element of type τ_i closest to the previous label (where closest means the greatest τ_i below the label, if $c <$ the label, and where closest means the least τ_i above the label, if c is above the label). If $a <$ some label, then the $l\tau \in (b_0, b_1)$ mentioned in the definition of the label is an element of type τ between b_0 and b_1. If c is in c_1, then the label mentions the least element of type τ_0 above this element of type τ, and by induction, $a >$ the label implies that c' and the rest of what defines the right labels will be found between this element of type τ and a. If c is in c_0, then the label mentions the greatest element of type τ_0 below this element of type τ, and by induction, $a >$ the label implies that c' and the rest of what defines the right labels will be found between this element of type τ and a. This ends the proof of the lemma.

Now by induction on subsets σ of $k-2$, we can locate each element of $I^{\text{right}}_{\{i:i<k-2\}}(\{d \in \lambda : d < a\})$ within the ordering $I^{\text{left}}_{\{i:i<k-2\}}(\{d \in \lambda : d < a\})$ and determine the set of $\equiv^{\text{loc}}_{k-2}$ classes between those left and right assignments of labels. By the induction hypothesis, we know the $\equiv^{\text{loc}}_{k-2}$ classes realized between any elements of $I^{\text{left}}_{\{i:i<k-2\}}(\{d \in \lambda : d < a\})$ and of $I^{\text{right}}_{\sigma_0}(\{d \in \lambda : d < a\})$, for σ_0 the lexicographical predecessor of σ. So, a fortiori, we know the $\equiv^{\text{loc}}_{k-2}$ classes realized, for $n \leq k-2$, since $\equiv^{\text{loc}}_{k-2}$ refines $\equiv^{\text{loc}}_m$. This allows us to compare, in any interval (b, c), the least τ and the greatest element of some other $\equiv^{\text{loc}}_{k-2}$ class – the least τ precedes the greatest τ_1 just in case something of type τ is realized between b and the greatest τ_1. Further, when a right label lies between

left labels b and c, we can determine which $\equiv^{\mathrm{loc}}_{k-2}$ classes are realized between $\sup b$ and the right label and which $\equiv^{\mathrm{loc}}_{k-2}$ classes are realized between the right label and c. This ends the proof of theorem 3.3.2. □

Now we prove theorem 3.0.2: That $\equiv_k$ implies identical trees of labels, identical orderings A on them, and identical $\equiv^{\mathrm{loc}}_{k-1}$ sequences on A and A^+ follows from theorem 3.3.1. That $\equiv_k$ holds when these data are identical follows from theorem 3.3.2, since if the data are identical, then player II can choose to play so as to respect $I_{\{i:i<k-1\}}(\lambda_i)$ and $\equiv^{\mathrm{loc}}_{k-1}$. Then, by theorem 3.3.2, data sufficient to prove $\equiv_{k-1}$ will be identical on either side of the played elements $a_i \in \lambda_i$, since this data depends only on the location of a_i within the $I_{\{i:i<k-1\}}(\lambda_i)$ and on the $\equiv^{\mathrm{loc}}_{k-1}$ class of (λ, a_i).□

3.4 Effective decision procedures and completions

Searching naïvely for a witness W as in theorem 3.0.3 is not effective at deciding the $\equiv_3$ or $\equiv_4$ classes of linear orders, since there are $6E14$ sets of pairs of $\equiv_2$ classes and $4E23769$ sets of pairs of $\equiv_3$ classes; the number of potential witnesses is the power of those sets. We would do better to search for the data of theorem 3.0.2. Then we will need an alternate game, a *local consistency game* to determine which datasets are consistent.

If U is a set of $\equiv^{\mathrm{loc}}_{k-1}$ classes, with or without an additional single $\equiv^{\mathrm{loc}}_{k-1}$ class on the left, and with or without an additional single $\equiv^{\mathrm{loc}}_{k-1}$ class on the right, and V is likewise, then we define $U+V$ just in case the same $\equiv^{\mathrm{loc}}_{k-1}$ class τ is on the right of U and on the left of V, or there is no $\equiv^{\mathrm{loc}}_{k-1}$ class on the right of U or on the left of V. If that holds, then let $U+V = U \cup V \cup \{\tau\}$ or $U+V = U \cup V$ if there is no $\equiv^{\mathrm{loc}}_{k-1}$ class on the right of U or on the left of V, and $U+V$ has on the left the $\equiv^{\mathrm{loc}}_{k-1}$ class that U has on the left (if any), and $U+V$ has on the right the $\equiv^{\mathrm{loc}}_{k-1}$ class that V has on the right (if any).

Definition 3.4.1: The *local consistency game* is the following: On each turn, the game state is a finite sequence $(c_i : i \leq n)$ of constants, a sequence $(m_i : i \leq n)$ of markers that either mark c_i as a cut or mark c_i with an $\equiv^{\mathrm{loc}}_{k-1}$ class, and a sequence $(U_i : i < n)$ of sets of $\equiv^{\mathrm{loc}}_{k-1}$ classes. The first player has two types of moves:

- Player I chooses $i < n$ and an element m of U_i. The second player then adds a new constant c left of c_i and right of c_{i-1}, with mark m an $\equiv^{\mathrm{loc}}_{k-1}$ class, and chooses two sets V_0 and V_1 such that $V_0 + V_1 = U_i$ such that V_0 has the element m on the right and V_1 has the element m on the left. The game state is then the finite sequence of $n+1$-many elements and $n+2$-many sets obtained by marking c with m and replacing U_i by the pair V_0, V_1.

- Player I chooses $i \leq n$ and a label in $I^{\mathrm{loc}}_{\{i:i<k-1\}}(m_c)$, say, to the right of c_i, such that every label on which this label depends has already been played. Player II chooses a constant $c > c_i$ (c may exist already or not; c may by necessity be beyond all c_i, especially if k is large and $(U_i : i < n)$ is small – either in that it is a short sequence or in that its elements U_i are small). The constant c is marked with an $\equiv^{\mathrm{loc}}_{k-1}$ class just in case the label describes the least or greatest τ, and not the unrealized infimum or

supremum of an infinite sequence of elements of type τ – player II intends that in the resulting linear order, c will be where that label, relative to c_i, must be assigned. Otherwise, the constant c is marked as a cut. Player II chooses new sets V_0, V_1 such that if V_0 is the set between c_i and the new constant c, $\{Th^{\text{loc}}_{k-2}(\rho) : \rho \in V_0\}$ is the set of $\equiv^{\text{loc}}_{k-2}$ classes which m_c says exists below the label.

The conditions on whether player II has survived grow as the game progresses. They are a set of conditions of two types: 1. that between certain constants c and d, all U_i ever created must omit $\equiv^{\text{loc}}_{k-1}$ class τ, or 2. that for a certain constant c, every interval immediately to the right of c must realize $\equiv^{\text{loc}}_{k-1}$ class τ. A new condition is created every time player I plays a move of the second type. If the label is $l\tau_0 \in (b, c)$ or $d\tau_0 \in (b, c)$, then the omission condition is that τ_0 is never realized between b and the new constant. If the label is $g\tau_0 \in (b, c)$ or $a\tau_0 \in (b, c)$, then the omission condition is that τ_0 is never realized between the new constant and c. If the label is $a\tau_0 \in (b, c)$, then we require that every set ever created immediately below the new constant realizes τ_0. If the label is $d\tau_0 \in (b, c)$ then we require that every set ever created immediately above the new constant realizes τ_0. Player II survives this move if all the conditions developed so far are met.

The initial state has any number of constants with any markings, and sets U_i. For instance, the initial state could be the data of theorem 3.0.2, which determine a general $\equiv_k$ class. Or the initial state could be a single set U_0 with an element on the left or not, and an element on the right or not. If the initial state has a single set U_0, then there is an initial constant c_0 on the left; it is marked with $\equiv^{\text{loc}}_{k-1}$ class τ just in case U_0 has τ on the left. There is an initial constant c_1 on the right and the $\equiv^{\text{loc}}_{k-1}$ class U_0 has on the right is m_1 (nothing, or an $\equiv^{\text{loc}}_{k-1}$ class).

We say that player I plays an *exhaustive* strategy if player I mentions every $\equiv^{\text{loc}}_{k-1}$ class in every interval U_i ever created, and mentions every label in $I^{\text{loc}}_{\{j:j<k-1\}}(m_i)$ for every constant c_i ever created which is marked with an $\equiv^{\text{loc}}_{k-1}$ class and not marked as a cut. If player II has a winning strategy in the consistency game and if player I plays an exhaustive strategy, then the set of constants which are not marked as cuts order grows into a model of the local classes in the initial state - a linear order λ with an element for each initial constant c_i for $i \leq n$ which is not marked as a cut, and which realizes exactly the local types in U_i for each $i < n$. On the one hand, every constant c_i marked with an $\equiv^{\text{loc}}_{k-1}$ class eventually realizes that $\equiv^{\text{loc}}_{k-1}$ class, as player I exhausts the labels in $I^{\text{loc}}_{\{j:j<k-1\}}(m_i)$ and marks them accordingly and player II can't violate m_i without losing. On the other hand, every element τ in every set U_i ever created corresponds to some constant c, because player I has exhausted each set U_i.

The consistency of a set U_0 of $\equiv^{\text{loc}}_{k-1}$ classes can be decided quickly, since the conditions on consistency are that for each $\tau_i \in U_0$ which is postulated to exist, the labels in $I^{\text{loc}}_{\{j:j<k-1\}}(\tau_i)$ have, in turn, fuller descriptions as $\equiv^{\text{loc}}_{k-1}$ classes. The set of things realized between a constant c_i and its neighbors is a set $\subseteq U_0$ which omits some type, and hence is a strict subset of U_0. In this way, we reduce the question of whether U_0 is consistent to a myriad of questions about whether smaller sets are consistent. Once those smaller questions are solved, call $H_{(\tau_0,U,\text{label})}$ the set of formulas $\phi_{(\rho,V)}$ which expresses that the set V of

local types is realized, and ρ is realized at the end, for each pair (ρ, V) such that $\rho \in U_0$ could extend a particular label in $I^{\mathrm{loc}}_{\{j:j<k-1\}}(m_i)$ and V could be the set of $\equiv^{\mathrm{loc}}_{k-1}$ classes realized between c_i and its label ρ. U_0 is consistent just in case the Horn theory

$$\bigwedge_{\tau\in U_0} \bigwedge_{(\mathrm{label}\in I^{\mathrm{loc}}_{\{j:j<k-1\}}(\tau_0))} (\tau(x) \rightarrow (\bigvee_{H_{(\tau_0,U,\mathrm{label})}} \phi_{(\rho,V)}))$$

is consistent. Satisfying that Horn theory means constructing strings of $\equiv^{\mathrm{loc}}_{k-1}$ classes until the local labels of each $\equiv^{\mathrm{loc}}_{k-1}$ class is satisfied. However, as in the linear consistency game of theorem 3.0.3, we can stop satisfying local labels as soon as the satisfaction process becomes repetitive. The resulting finite structures which show how to string local classes together in a model λ are the following:

Definition 3.4.2: An almost locally closed set is any nonempty $A \subseteq \lambda \cup \lambda^+$ such that for each $a \in A$ there is some $a_0 \in A$ such that $(\lambda, a) \equiv_{k-1} (\lambda, a_0)$ and there is a homomorphism h from the ordered set $I^{\mathrm{loc}}_{\{i:i<k-1\}}(\lambda, a_0)$ into A sending $l\tau \in (b, c)$ to the least element between $h(b)$ and $h(c)$ of $\equiv^{\mathrm{loc}}_{k-1}$ class τ, and sending $d\tau \in (b, c)$ to the greatest cut (e, f) in λ^+ such that f contains every element between $\sup b$ and $\inf c$ of $\equiv^{\mathrm{loc}}_{k-1}$ class τ, and likewise for $g\tau \in (b, c)$ and $a\tau \in (b, c)$. For each label $d\tau \in (b, c)$ or $a\tau \in (b, c)$ of $I^{\mathrm{loc}}_{\{i:i<k-1\}}(\lambda, a_0)$, A also contains an example: an element of type τ above $d\tau \in (b, c)$ (or an element of type τ below $a\tau \in (b, c)$) such that for any $g \in \lambda$ between the cut and the example, there is an $h \in A$ not between the example and the cut, such that $(\lambda, c, d, g) \equiv_{k-1} (\lambda, c, d, h)$. For each $\equiv^{\mathrm{loc}}_{k-2}$ class τ which $Th^{\mathrm{loc}}_{k-1}(\lambda, a_0)$ knows to exist between two labels, A contains an *example*: an element of type τ between h of those two labels.

Among the almost locally closed sets which contain an element with $\equiv^{\mathrm{loc}}_{k-1}$ class τ, we are especially interested in the minimal sets, i.e., those sets A for which there is no almost locally closed proper subset of A which also realizes τ.

These sets help us to define a finitely axiomatizable linear order in any $\equiv_k$ class. We call σ a *cut-state* if it contains a single set U_0 of $\equiv^{\mathrm{loc}}_{k-1}$ classes, with or without an $\equiv^{\mathrm{loc}}_{k-1}$ class on the left end and with or without an $\equiv^{\mathrm{loc}}_{k-1}$ class on the right end, since it is the state of the linear consistency game at a cut $(\{c_{i-1}\}, \{c_i\})$ in the set of constants. We call a cut-state σ *consistent* if player II can win the linear consistency game in which $\sigma = (c_0, c_1), (m_0, m_1), (U_0)$ is the initial condition. For each consistent cut-state σ we will write a sentence δ_σ which expresses that the elements of σ are realized densely. We might be tempted to insist that for all $x_0 \in \lambda$ and for all $x_1 \in \lambda($ if (λ, x_0, x_1) satisfies σ, then $(\lambda, x_0, x_1) \models \delta^{(x_0,x_1)}_\sigma)$. This invites a study of the consistency of a family $\{\delta_\sigma : \sigma \in W\}$ of complete sentences, on overlapping intervals. If the consistency of such a set of sentences can be decided, then we can define completions without $\equiv^{\mathrm{loc}}$ classes. But theorem 3.0.2 allows us to construct a generic $\equiv_k$ class of linear order in stages, as in the linear consistency game, while controlling only the local $\equiv^{\mathrm{loc}}_{k-1}$ classes in each gap, and not the set of pairs of $\equiv_k$ classes realized in each gap. Admittedly, the theory of linear order can be described, decided, and completed, using overlapping intervals as the basic building block. Instead, we

use local neighborhoods as the basic building block with which we construct, decide, and complete theories of linear order.

Definition 3.4.3: Suppose $x_0 \in \lambda, x_1 \in \lambda$ and there is an almost locally closed set $A \subseteq \lambda$ containing $\{x_0, x_1\} \subseteq A$ such that (λ, x_0, x_1) satisfies σ, and A is minimal among the almost locally closed sets containing $\{x_0, x_1\}$. Then let δ_σ be

$$(\exists x_a : a \in A, x_0 < a < x_1)$$

$$\bigwedge_{a,b \in A \cap \lambda, \text{adjacent}} \delta^{(x_a,x_b)}_{Th^{\text{loc}}_{k-1}(\lambda,x_0),\{Th^{\text{loc}}_{k-1}(\lambda,x):x_0<x<x_1\},Th^{\text{loc}}_{k-1}(\lambda,x_1)}.$$

If σ indicates a non-empty set of $\equiv^{\text{loc}}_{k-1}$ classes between x_0 and x_1, then for τ the $\equiv^{\text{loc}}_m$ class of any element of σ, the label $l\tau > x_0$ or $d\tau > x_0$ will be assigned below x_1. On the other hand, each element of A is defined in relation to some other element of A, so that in each gap between elements of A, some part of σ is not realized. Therefore we have defined δ_σ in terms of $\{\delta_\rho : \rho \text{ is a proper subset of } \sigma\}$.

Definition 3.4.4: Suppose σ is not satisfied within one minimal almost locally closed class, as was the case in the preceding definition. Suppose $x_0 \in \lambda$ and $y_1 \in \lambda^+$ (and consider the possibilities, too, that the left end is a cut or the right end is an element) and that (λ, x_0, y_1) satisfies the cut-state σ. Since $x_0 \in \lambda$, find an almost locally closed set $A_0 \subseteq \lambda$ containing x_0. Let δ_σ assert the existence of A_0:

$$(\exists x_a : a \in A_0 \cap \lambda, x_0 < a < y_1)($$

$$\bigwedge_{a,b \in A_0 \cap \lambda, \text{adjacent}} \delta^{(x_a,x_b)}_{Th^{\text{loc}}_{k-1}(\lambda,x_a),\{Th^{\text{loc}}_{k-1}(\lambda,x):x_a<x<x_b\},Th^{\text{loc}}_{k-1}(\lambda,x_b)})$$

Let σ_{-A_0} be a set of almost locally closed sets (subsets of λ) such that σ requires the existence of exactly the $\equiv^{\text{loc}}_{k-1}$ classes $\cup\{\{Th^{\text{loc}}_{k-1}(\lambda, a) : a \in A\} : A \in \sigma_{-A_0}\} \cup \{Th^{\text{loc}}_{k-1}(\lambda, a) : a \in A_0, x_0 < a\}$. Let δ_σ assert that every x in (x_0, y_1) is $\equiv_{k-1}$ to a first move made in A_0, or is part of an almost locally closed set $\equiv_{k-1}$ to something in σ_{-A_0} :

$$\forall x((x_0 < x < y_1) \rightarrow (\exists z((\vee_{b \in A_0} z < x_b) \wedge (\lambda, x_0, x, y_1) \equiv_{k-1} (\lambda, x_0, z, y_1)))\vee$$

$$(\vee_{A \in \sigma_{-A_0}}(\exists x_a : a \in A \cap \lambda)(\vee_{a \in A \cap \lambda} x_a = x)\wedge$$

$$\bigwedge_{a,b \in A \cap \lambda, \text{adjacent}} \delta^{(x_a,x_b)}_{Th^{\text{loc}}_{k-1}(\lambda,x_a),\{Th^{\text{loc}}_{k-1}(\lambda,x):x_a<x<x_b\},Th^{\text{loc}}_{k-1}(\lambda,x_b)}))$$

Let δ_σ further assert that above A_0 the elements of σ_{-A_0} are realized without either upper or lower bound, so that every element is $\equiv_{k-1}$ to a first move in A_0 or all possible elements of σ_{-A_0} are realized below it, and that, without condition (since $y_0 \in \lambda^+$), all elements of σ_{-A_0} are realized above it:

$$\forall x((x_0 < x < y_1) \rightarrow ($$

$$((\exists z((\vee_{b \in A_0} z < x_b) \wedge (\lambda, x_0, x, y_1) \equiv_{k-1} (\lambda, x_0, z, y_1)))\vee$$

$$(\wedge_{A \in \sigma_{-A_0}}((\exists x_a : a \in A \cap \lambda)(x_0 < x_a < x)\wedge$$

$$\bigwedge_{a,b\in A\cap\lambda,\text{adjacent}} \delta^{(x_a,x_b)}_{Th^{\mathrm{loc}}_{k-1}(\lambda,x_a),\{Th^{\mathrm{loc}}_{k-1}(\lambda,x):x_a<x<x_b\},Th^{\mathrm{loc}}_{k-1}(\lambda,x_b)})))$$

$$\wedge(\wedge_{A\in\sigma_{-A_0}}((\exists x_a : a\in A\cap\lambda)(x<x_a<y_1)\wedge$$

$$\bigwedge_{a,b\in A\cap\lambda,\text{adjacent}} \delta^{(x_a,x_b)}_{Th^{\mathrm{loc}}_{k-1}(\lambda,x_a),\{Th^{\mathrm{loc}}_{k-1}(\lambda,x):x_a<x<x_b\},Th^{\mathrm{loc}}_{k-1}(\lambda,x_b)})))$$

Let δ_σ further require that the elements of σ_{-A_0} are realized densely – for every pair of elements $x<y$, if x is not $\equiv_{k-1}$ to a first move played in A_0 and such that $\{x,y\}$ is not spanned by a single element of σ_{-A_0} then every element of σ_{-A_0} is realized between x and y:

$$\forall x\forall y((x_0<x<y<y_1)\rightarrow($$

$$(\exists z((\vee_{b\in A_0} z<x_b)\wedge((\lambda,x_0,x,y_1)\equiv_{k-1}(\lambda,x_0,z,y_1))))$$

$$\vee\vee_{A\in\sigma_{-A_0}}((\exists x_a : a\in A\cap\lambda)$$

$$((\bigwedge_{a,b\in A\cap\lambda,\text{adjacent}} \delta^{(x_a,x_b)}_{Th^{\mathrm{loc}}_{k-1}(\lambda,x_a),\{Th^{\mathrm{loc}}_{k-1}(\lambda,x):x_a<x<x_b\},Th^{\mathrm{loc}}_{k-1}(\lambda,x_b)})$$

$$\wedge(\vee_{a\in A\cap\lambda}x_a=x)\wedge(\vee_{a\in A\cap\lambda}x_a=y)))$$

$$\vee(\wedge_{A\in\sigma_{-A_0}}((\exists x_a : a\in A\cap\lambda)((x<x_a<y)\wedge$$

$$\bigwedge_{a,b\in A\cap\lambda,\text{adjacent}} \delta^{(x_a,x_b)}_{Th^{\mathrm{loc}}_{k-1}(\lambda,x_a),\{Th^{\mathrm{loc}}_{k-1}(\lambda,x):x_a<x<x_b\},Th^{\mathrm{loc}}_{k-1}(\lambda,x_b)})))))).$$

The conjunction of the foregoing four sentences is what we call δ_σ.

If σ indicates that no $\equiv^{\mathrm{loc}}_{k-1}$ classes are realized between x_0 and x_1, then $\delta_\sigma = Th_k(\emptyset)\wedge$ some information about the local class of the left end (if there is one) to the right of the interval, and information about the local class of the right end (if there is one) to the left of the interval. E.g., if there is an $\equiv^{\mathrm{loc}}_{k-1}$ class on the left and an $\equiv^{\mathrm{loc}}_{k-1}$ class on the right, but σ is empty, then these two classes can be realized at a pair of immediate predecessor and successor. The formula δ_σ is then much like an $\equiv^{\mathrm{loc}}_{k-1}$ class, in that it determines an $\equiv^{\mathrm{left}}_{k-1}$ class right of the pair, and an $\equiv^{\mathrm{right}}_{k-1}$ class left of the pair. If σ has no $\equiv^{\mathrm{loc}}_{k-1}$ class on the left or on the right and is empty, then δ_σ is $Th_k(\emptyset)$ since the second sentence in the definition above says that every x is part of an almost locally closed set in σ_{-A_0}: $\forall x((y_0<x<y_1)\rightarrow\vee\emptyset)$ is $Th(\emptyset)$.

Now we define, by induction again on σ, a model of δ_σ. Suppose that models of δ_ρ exist whenever ρ is a proper subset of σ.

Definition 3.4.5: If $A\subseteq\lambda$ is an almost locally closed set, minimal among those realizing a particular $\equiv^{\mathrm{loc}}_{k-1}$ class, then let μ_A contain $A\cap\lambda$ and a copy of λ_ρ in every cut in A^+ (note that we include all of A, not only $A\cap\lambda$, on purpose) in which the set ρ is realized, perhaps with an $\equiv^{\mathrm{loc}}_{k-1}$ class on the left or an $\equiv^{\mathrm{loc}}_{k-1}$ class on the right. Let the least and greatest elements of A be a_0 and a_1. By the definition of an almost locally closed set, there exist $b_0,b_1\in A$ such that $(\lambda,a_i)\equiv_{k-1}(\lambda,b_i)$. Let the half-open interval in μ_A between a_i and b_i, including a_i and not b_i, be μ_i. Let $\lambda_A=\mu_0\times\omega^*+\mu_A+\mu_1\times\omega$. In this way we make out of an almost locally closed set an interval for the linear order λ_A.

Suppose σ is as in definition 3.4.3, i.e., that there is a single almost locally closed set A such that if σ describes an $\equiv^{\mathrm{loc}}_{k-1}$ class on the left, there is some $x_0 \in A$ such that (λ_A, x_0) is in that class, and if σ describes an $\equiv^{\mathrm{loc}}_{k-1}$ class on the right, there is some $x_1 \in A$ such that (λ_A, x_1) is in that class, and such that between x_0 (or $x0 = (\emptyset, \lambda_A)$, if σ does not describe an $\equiv^{\mathrm{loc}}_{k-1}$ class on the left) and x_1 (or $x_1 = (\lambda_A, \emptyset)$, if σ does not describe an $\equiv^{\mathrm{loc}}_{k-1}$ class on the right), (λ_A, x_0, x_1) satisfies σ, and such that A is minimal among all almost locally closed sets containing two elements x_0, x_1 of the given $\equiv^{\mathrm{loc}}_{k-1}$ classes. Then choose that $A \subseteq \lambda$ and elements x_0, x_1 for which 3.4.3 defines the density formula δ_σ and add to λ_σ constants for elements of A realizing the $\equiv^{\mathrm{loc}}_{k-1}$ class that σ describes on the left, if there is one, and the $\equiv^{\mathrm{loc}}_{k-1}$ class that σ describes on the right, if there is one. For instance, if σ describes neither a left nor a right element, then $\lambda_\delta = \lambda_A$.

Suppose otherwise, i.e., that no one almost locally closed set A, minimal among those that realize two $\equiv^{\mathrm{loc}}_{k-1}$ classes, realizes every $\equiv^{\mathrm{loc}}_{k-1}$ class described by δ. If σ has the $\equiv^{\mathrm{loc}}_{k-1}$ class $|tau$ on the left and no $\equiv^{\mathrm{loc}}_{k-1}$ class on the right, let A_0 be the minimal almost locally closed set containing an element $a_0 \in A_0$ of type σ as chosen in definition 3.4.4, let σ_{-A_0} be the set of almost locally closed sets chosen in definition 3.4.4, and let λ_δ be $(\lambda_A, a_0)+$ a dense shuffle of $\{\lambda_A : A \in \sigma_{-A_0}\}$.

Theorem 3.4.1: If σ is a set of $\equiv^{\mathrm{loc}}_{k-1}$ classes with or without a single $\equiv^{\mathrm{loc}}_{k-1}$ class on the left and with or without a single $\equiv^{\mathrm{loc}}_{k-1}$ class on the right, λ_σ is a linear order with one ore more constants which satisfies δ_σ, with the constants satisfying δ_σ's single type on the left or right; the remaining $\equiv^{\mathrm{loc}}_{k-1}$ classes are realized between these constants.

Proof: If σ is satisfied within a single linear order λ_A, for A a single almost locally closed set, then λ_σ was chosen in the previous definition's first paragraph to satisfy σ. If σ is not satisfied within a single λ_A, then δ_σ requires that the $\equiv^{\mathrm{loc}}_{k-1}$ class on the left is part of a minimal almost locally closed set A_0 – the second paragraph of the previous definition defines λ_{A_0} – and δ_σ requires that the δ_σ requires something similar on the right. Finally, δ_σ requires that almost-locally closed sets in σ_{-A_0} be densely ordered. In the last paragraph of the previous definition, we find that λ_δ does in fact densely order $\{\lambda_A : A \in \sigma_{-A_0}\}$. It remains to check that in each interval in each λ_A, between any a_0 and $a_1 \in A$, such that the set ρ of $\equiv^{\mathrm{loc}}_{k-1}$ classes is realized between a_0 and a_1, then λ_ρ, with constants for a_0 and a_1 if they are in λ and not λ^+, satisfies δ_ρ with the constants a_i interpreting whatever $\equiv^{\mathrm{loc}}_{k-1}$ classes ρ requires on the left and right. By induction on strict subsets of σ, we may assume this is true. As the base case, if σ is empty, then δ_ρ describes the empty set and λ_ρ is the empty set. □

By theorem 3.0.2, $\mu_0 \equiv_k \mu_1$ holds just in case μ_0 and μ_1 have certain data in common. That data form an initial state of the local consistency game. So, for any $\equiv_k$ set, we take this initial state, extend its $\equiv^{\mathrm{loc}}_{k-1}$ classes to almost locally closed sets, and write δ_σ. For any finite number k, for any linear order λ, the $\equiv_k$ class of λ is determined by one such initial state for the local consistency game. For such a state, we form λ_σ in each interval, for $\sigma = U_i$, and by cutting the models λ_σ at the constants which refer to the locations of the $\equiv^{\mathrm{loc}}_{k-1}$ classes on the left of U_i and the right of U_{i-1}, we form a model of the entire initial state. A subsequence $(c_i : i_0 < i < i_1)$ of the constants may be close, in that the sets

U_i between them is small. Then it is likely that a single almost locally closed set which is minimal among almost locally closed sets containing even c_{i_0} will contain them all. We could then form the single set λ_A to explain the whole sequence $(c_{i_0}, U_{i_0}, c_{i_0+1}, \ldots c_{i_1})$, though the theoretically simpler definition is simply to form λ_A on each triple c_i, U_i, c_{i+1}, to cut it at the constants for c_i and c_{i+1}, and to add these linear orders together. In any case, wherever there are constants $c_i < c_{i+1}$ such that no one almost locally closed set which is minimal among those containing elements with the $\equiv^{\rm loc}_{k-1}$ classes of c_i and c_{i+1} realizes all of U_i, then we choose an almost locally closed set A_i which is minimal among those containing an element with the $\equiv^{\rm loc}_{k-1}$ class of c_i and an almost locally closed set A_{i+1} which is minimal among those containing an element with the $\equiv^{\rm loc}_{k-1}$ class of c_{i+1} and realize the $\equiv^{\rm loc}_{k-1}$ classes left over from U_i densely between (λ_{A_i}, c_i) and $(\lambda_{A_{i+1}}, c_{i+1})$. By the following theorem, we are justified in calling the linear order which is the piecewise sum, over constants $(c_i, i \leq n)$, of dense linear orders λ_δ, the *piecewise-dense* model $PWD(Th_k(\lambda))$ of $Th_k(\lambda)$.

Theorem 3.4.2: Let k be any finite number; let λ be any linear order. Then the followin equivalence holds: $\lambda \equiv_k PWD(Th_k(\lambda))$ and $PWD(Th_k(\lambda))$ is finitely axiomatized by

$$Th_k(\lambda) \wedge \bigwedge_{a,b \in I_{\{i:i<k-1\}}(\lambda), \text{adjacent}} \delta^{(a,b)}_{\{Th^{\rm loc}_{k-1}(\lambda,x):a<x<b\}}$$

where instead of postulating the existence of elements x_a for each a in the set of indices $I_{\{i:i<k-1\}}(\lambda)$ and restricting δ to occur between x_a and x_b, we instead use the fact that a and b are definable elements and cuts in λ, and we restrict δ to occur among the set of elements $d \in \lambda$ such that d is above the defined cut a and below the defined cut b.

Proof: As in the previous theorem, $PWD(Th_k(\lambda))$ satisfies δ_σ for each adjacent pair of elements $a, b \in I_{\{i:i<k-1\}}(\lambda)$, where δ describes the $\equiv^{\rm loc}_{k-1}$ class classes realized between the defined elements or cuts a and b. That is, Th_k of the interval between a and b only requires that the local classes in σ be realized between the definable elements or cuts a and b. The formula $\delta^{(a,b)}_\sigma$ adds to this a choice about how the local classes form into almost locally closed sets, insists that this happens regularly (uniformly within the interval (a, b)), and insists that these almost locally closed sets are realized densely. But since this certainly implies that exactly the $\equiv^{\rm loc}_{k-1}$ classes in δ are realized, for any first move played in (a, b) in λ or $PWD(Th_k(\lambda))$, player II can answer with an element which has its $\equiv^{\rm loc}_{k-1}$ class in δ and which is realized between a and b. This implies by theorem 3.3.2 that the linear orders left and right of the played elements are $\equiv_{k-1}$. To see that $\lambda \equiv_k PWD(Th_k(\lambda))$ is finitely axiomatized by the given formula, we will compute its $\equiv_{k+m}$ class from the given formula, for any natural number m. By theorem 3.0.2, the $\equiv_{k+m}$ class is determined by sequences $(Th^{\rm loc}_{k+m-1}(\lambda, a) : a \in I_{\{i:i<k+m-1\}}(\lambda) \cap \lambda)$ and $(\{Th^{\rm loc}_{k+m-1}(\lambda, a) : b < a < c\} : (b, c) \in (I_{\{i:i<k+m-1\}}(\lambda))^+)$. For each natural number m, $I_{\{k+m\}}(\lambda)$ adds labels for the first and last occurrence of each $\equiv^{\rm loc}_{k+m}$ class. Those $\equiv^{\rm loc}_{k+m}$ classes which are realized in λ_{A_i} for A_i the almost locally closed element containing the i-th element of $I_{\{i:i<k-1\}}(\lambda)$, are realized as λ_A orders them – by induction on proper subsets $\rho \subseteq \sigma$, the intervals in λ_A determine where these $\equiv^{\rm loc}_{k+m}$ classes begin and end. The remaining $\equiv^{\rm loc}_{k+m}$

classes which are realized in $PWD(Th_k(\lambda))$ are realized in λ_A for $A \in \sigma_{-A_0}$ (or $A \in \sigma_{-A_0,-A_1}$), sets of almost locally closed sets which realize all $\equiv^{\rm loc}_{k-1}$ classes not realized in A_0 (or not realized in A_0 or in A_1). Again, by induction on proper subsets of σ, we can determine which $\equiv^{\rm loc}_{k+m}$ classes are realized in λ_A. It is easy to see that density will force these $\equiv^{\rm loc}_{k+m}$ classes to be realized without lower or upper bound and to be realized all the way down to the cut (λ_{A_0}, the shuffle of σ_{-A_0}). □

3.5 Semimodels

Unlike previous sections, this section describes a topic without the motivation of a main theorem to which everything trends. Instead we gather together some results on *semimodels*, a rich concept.

Theorem 3.0.3 shows that for any consistent set U of pairs of $\equiv_k$ classes, a finite set W of information witnesses the consistency of U. The proof of theorem 3.0.3 adds this twist: if player I plays an exhaustive strategy in the linear consistency game in definition 3.1.1, and if player II plays according to a function (f_0, f_1) from W to $W \times W$, then other variations in player I's strategy have no effect on the linear order λ which is created during the game – each element of λ is waiting to be created because W defines it in terms of other elements of λ which are waiting to be created, and the order in which player I goes about turning these into played constants does not change the set of elements which are ultimately created, nor its ordering. That is, "λ is built in stages according to $W, (f_0, f_1)$" is a complete description of λ. However, we went ahead and defined $PWD(Th_k(\lambda))$ and proved in theorem 3.4.2 that this λ has a complete description in first-order logic over the vocabulary $<$. The simpler definition "λ is built in stages according to $W, (f_0, f_1)$" can be expressed over the vocabulary $<$ in a logic which is first-order and has the additional capacity that it recognizes "stages."

Given any linear order λ, for each pair x_0, x_1 of elements of λ, let $U(\lambda, x_0, x_1) = U(Th_{k+1}(\{x_2 \in \lambda : x_0 < x_2 < x_1\}))$ be the set of pairs of $\equiv_k$ classes realized as $(Th_k(x_0, x_2), Th_k(x_2, x_1))$ for various x_2 between x_0 and x_1. The relationship between $W, (f_0, f_1)$ and the $\equiv_{k+1}$ class $U(\lambda)$ of λ is complicated. If it were possible to interpret each complete description $W, (f_0, f_1)$ as a set $U(\lambda)$, we would have a complete theory in the given $\equiv_{k+1}$ class. Theorem 3.0.3 checks that $Th_{k+1}(\lambda)$ is consistent by finding that for any constants x_0 and x_1 which are defined and adjacent at some stage, those constants define an interval (x_0, x_1) (similarly, one constant which was at some stage the least defined constant defines intervals with no left endpoint or no right endpoint, and no constants define the entire linear order itself) such that $U(x_0, x_1)$ is an element of a set $W = \{U_I : I \text{ is an interval in } \lambda\}$ so that for $U \in W$ and for any of U's elements, e.g., $(Th_k(x_0, x_2), Th_k(x_2, x_1)) \in U_{(x_0,x_1)}$, there exists a pair (V_0, V_1) of elements of W such that $(\xi(V_0), \xi(V_1)) = (Th_k(x_0, x_2), Th_k(x_2, x_1))$ and $V_0 + (\emptyset, \emptyset) + V_1 = U_{(x_0,x_1)}$. In λ, there must exist an element x_2 between x_0 and x_1 such that $Th_k(x_0, x_2) = \xi(V_0)$ and $Th_k(x_2, x_1) = \xi(V_1)$. There must exist Skolem functions f_0 and f_1 for the formula $\forall U \in W(\forall(\phi, \psi) \in U(\exists V_0 \exists V_1(((\xi(V_0), \xi(V_1)) = (\phi, \psi)) \wedge (V_0 + (\emptyset, \emptyset) + V_1 = U_{(x_0,x_1)}))))$, which expresses consistency. But for triples $x_0, x_1, x_2 \in \lambda$ such that x_0 and x_1 were never adjacent during the construction of λ, x_0 and x_1 can have the same set

$U_{(x_0,x_1)}$ and the triples x_0, x_1, x_2 can have the same $(Th_k(x_0,x_2), Th_k(x_2,x_1))$, while the pair $(U(x_0,x_2), U(x_2,x_1))$ may be very different from the value

$$(f_0(U_{(x_0,x_1)}, (Th_k(x_0,x_2), Th_k(x_2,x_1))), f_1(U_{(x_0,x_1)}, (Th_k(x_0,x_2), Th_k(x_2,x_1))))$$

of those Skolem functions! If λ has been created in stages according to the functions (f_0, f_1) from W to $W \times W$, then λ has some triples which obey (f_0, f_1) (at least, those triples x_0, x_1, x_2 for which x_0 and x_1 were at one point adjacent), but λ might well have triples which do not obey (f_0, f_1). That every triple in λ obeys $W, (f_0, f_1)$ can be expressed in first-order logic as $\forall x_0(\forall x_1(\chi))$ where χ is the formula:

$$\wedge_{U \in W}(\sigma_U^{(x_0,x_1)} \rightarrow (\forall x(\wedge_{(\phi,\psi)\in U}((\phi^{(x_0,x)} \wedge \psi^{(x,x2)}) \rightarrow$$

$$((f_0(U,(\phi,\psi)))^{(x_0,x)} \wedge (f_1(U,(\phi,\psi)))^{(x,x_1)})))))$$

If this formula is consistent, it finitely axiomatizes any model λ. For we could compute Th_{k+m} of any interval (x_0, x_1) in λ as σ_Q where Q is the set of pairs of Th_{k+m-1} theories of intervals (x_0, x) and (x, x_1) for various x in the interval (x_0, x_1).

That is, if player II has a winning strategy in the consistency game and plays that strategy as a function, and if that strategy turns out to hold of all triples the resulting linear order is complete. Every linear order can result from player II's play in the consistency game, so long as player II adds some randomness to the strategy. For some initial states U_0 there are sets W which prove that U is consistent, so there is always a function f which constructs a linear order. But in general, there are pairs x_0, x_1 which arise in the tree of constants constructed during play which were never neighbors during the game, and yet which, at the end of the game, have the same state $U_{(x_0,x_1)}$ as some pair y_0, y_1 which were neighbors during the game. How can we describe the linear order which is built during the consistency game in which player II plays a strategy which is a function? semimodels are a good way to describe, up to $\equiv$, models which are built according to repeated rules, because they address the notion of "stages."

Definition 3.5.1: ([3]) A semimodel is a nested sequence $(M_i : i < \omega)$ of finite sets with a common ordering on $\cup_{i<\omega} M_i$. We call $(M_i : i < k)$ the rank k part of M. We say $M \models^{\text{semi}} \phi$ if $(M_i : i < \omega) \models \phi^{\text{semi}}$, where ϕ^{semi} is the relativization of ϕ in which we replace any subformula $\exists x\psi$ of ϕ which occurs within the scope of n-many quantifiers by $\exists x((x \in M_n) \wedge \psi)$ and we replace any subformula $\forall x\psi$ of ϕ which occurs within the scope of n-many quantifiers by $\forall x((x \in M_n) \rightarrow \psi)$.

$$\text{Let } \chi_1 = ((\forall x_0 \forall x_1 \chi) \wedge (\forall x_1 \forall x_0 \chi)).$$

Now $\{\chi_1, \exists y_0(\chi_1), \exists y_0(\exists y_1(\chi_1)), \dots\}$, the set containing χ_1 with any number of dummy quantifiers prepended in front of χ_1, describes up to $\cong$ the countable semimodel built by iterating the Skolem functions f_0 and f_1. Thus, every $\equiv_k$ class contains a linear order with a simple semimodel description. semimodels were introduced with the following theorem in mind:

Theorem 3.5.1: ([3]) If U is a class of finite semimodels, and the following hold:

1. U is recursively enumerable,
2. For any formula ϕ and any semimodel $L \in U$ such that the rank of L as a semimodel is the rank of ϕ as a formula, if $L \models^{\text{semi}} \phi$, then L extends to a full linear order λ, such that $\lambda \models \phi$,
3. For any formula ϕ and any linear order λ such that $\lambda \models \phi$, ϕ^{semi} holds in some semimodel $L \in U$,

then the theory of linear order is decidable.

Proof: Enumerate the implications of the theory of linear order, and look for $\neg\phi$. Meanwhile, enumerate elements of U, and look for $L \in U$ of rank equal to the quantifier rank of ϕ, such that $L \models \phi$. By condition 2, some linear order λ models ϕ, too. By condition 3, if ϕ is consistent, then this procedure terminates in the discovery of a semimodel of ϕ. □

The second condition rejects a number of intuitive semimodels, if the semimodels $(M_i : i < \omega)$ which are defined by repetitive play of a winning strategy in a consistency game are intuitive and if $(M_i : i < \omega)$ extends to $\cup_{i<\omega} M_i$. These structures are described up to $\cong$ in the class of countable semimodels, by a theory which affixes dummy quantifiers to the formula χ_1, given above. If there were a model of any element of that semi-theory, there is a model of the whole theory, since dummy variables dont alter whether a model satisfies a sentence, or not. The model would be finitely axiomatized by χ_1. For some winning strategies in the consistency game, there is no model of χ_1. These semimodels must be excluded from U. Even if χ_1 is not consistent, the semimodels $(M_i : i < \omega)$ have a consistent union and a simple semi-theory, even though the theory of the linear order $\cup_{i<\omega} M_i$ is not χ, i.e., for some natural number k, it holds, for many winning strategies in the consistency game, that $(M_i : i < k) \not\equiv_k^{\text{semi}} -1 \cup_{i<\omega} M_i$. We write the rank k part of a semimodel as the sequence $(s_i : i < n)$ where s_i is the least number n such that the i-th element of M_{k-1} is is M_n. From such a sequence we recover the semimodel's stages as $M_j = \{i : s_i \leq j\}$. We add semimodels by concatenating their sequences – i.e., we write one after the other. We multiply semimodels $N \times M$ by replacing every element of M of rank i by a copy of N in which every number has been increased by i. This usually produces a lot of waste which we can then trim away, finding a smaller sequence which is $\equiv_k$.

Lemma 3.5.1: If M is a semimodel and μ is a model, then the following are equivalent:

- For every sentence ϕ of quantifier rank k, $M \models^{\text{semi}} \phi$ just in case $\mu \models \phi$.
- Player II has a winning move in the EF game between M and μ, where on the j-th move, any move played in M must be played in M_j.

Proof: If there is a sentence ϕ violating the first item, then player I can use that as a winning strategy in item II. On the other hand, from a winning strategy for player I in item II we can create a formula ϕ which is satisfied in μ just in case it is not semimodel satisfied in M. □

If one or both of those conditions occur, we say $M \equiv_k^{\text{semi}} \mu$. The $\equiv_2$ classes of linear orders have semimodels:

$$\emptyset, 0, 00, 000, 100, 001, 101.$$

Lemma 3.5.2: $(0)^{2^{k-1}}(1)^{2^{k-2}}(2)^{2^{k-3}}\dots(k-1)^{2^0} \equiv_k^{\text{semi}} \omega$.

Proof: We play the EF game between the semimodel and ω – in the semimodel the j-th move is restricted to M_j. We answer a large natural number with the last 0. To the right, this leaves the statement of the lemma for $k-1$, which holds by induction. To the left, this leaves $(0)^{2^{k-1}-1}$ which is $\equiv_k^{\text{semi}}$ to any large, finite linear order, as the reader may wish to prove by induction. □

A sequence (s_i) is an $\equiv_k^{\text{semi}}$ semimodel for the integers, Z if 1. it tapers, from 0 to k, at least as slowly as in the preceding lemma, and 2. it is continuous - it never ascends from j to $j+2$ or descends from $j+2$ to j, without the value j in between.

Lemma 3.5.3: If $E \equiv_k^{\text{semi}} \eta$, the countable dense linear order without endpoints, then E with every element increased by one $+(0)+E$ with every element increased by one $\equiv_{k+1}^{\text{semi}} \eta$. $101 \equiv_2^{\text{semi}} \eta; 2120212 \equiv_3^{\text{semi}} \eta$.

Proof: Like E, η has only one type of element. If we assign the variable x to an element of that type, the k-quantifier theory of η left of x or right of x is the k-quantifier theory of η. Base case: $\emptyset \equiv_0^{\text{semi}} \eta$. □

To these examples, and the properties of $\times$ and $\sum$ for linear orders, we can also add semimodel versions of the random shuffle of a number of linear orders, and create a semimodel for any element of the hierarchy M_{LL} of [7], since that hierarchy is defined by $+$, $\times\omega$, $\times\omega^*$ and shuffle. But we don't find semimodels especially convenient for defining the shuffle – the resulting semimodel is very large, and repeats many sequences so that it's easy to define but unwieldy to work with. Instead, semimodels handle $\equiv_k^{\text{loc}}$ classes gracefully, and short semimodels in each $\equiv_k$ class can be obtained from the data of theorem 3.0.2. The following theorem allows us to write semimodels in each $\equiv_k$ class which are shorter than M_{LL} semimodels, and much shorter than those expressing the Skolem functions of theorem 3.0.3:

Theorem 3.5.2: For any linear order μ and any finite k, there is a semimodel M which is $\equiv_k^{\text{semi}}$ μ such that $|M_0| = |I_{\{i:i<k-1\}}(\lambda) \cap \lambda| + \sum\{|\{Th_{k+m-1}^{\text{loc}}(\lambda, a) : b < a < c\}| : (b,c) \in (I_{\{i:i<k+m-1\}}(\lambda))^+\}$ the size of M_k is bounded by $|M_0|$ times an upper bound on the size of the semimodels which express the various $\equiv_{k-1}^{\text{loc}}$ classes in μ.

Proof: We write semimodels for $\equiv_{k-1}^{\text{loc}}$ classes as follows: let 0 represent the element whose $\equiv_{k-1}^{\text{loc}}$ class we wish to describe, and add on either side add semimodels for $\equiv_{k-1}^{\text{left}}$ and $\equiv_{k-1}^{\text{right}}$ classes. An $\equiv_{k-1}^{\text{left}}$ class has as its semimodel any M such that each $\equiv_{k-1}$ class ϕ in the $\equiv_{k-1}^{\text{left}}$ class has a semimodel $M+N$. A model in which a set of local types exists can be obtained by simply concatenating semimodels for those local classes. By theorem 3.0.2, we know that an $\equiv_k$ class is equivalent to a sequence of elements with determined $\equiv_{k-1}^{\text{loc}}$ classes, and sets of $\equiv_{k-1}^{\text{loc}}$ classes between them. The concatenation of a sequence of semimodels, one for each label and one for each $\equiv_{k-1}^{\text{loc}}$ class supposed to exist between the labels $\models^{\text{semi}}$ the desired $\equiv_k$ class. □

The inextensible $\equiv_2^{\text{loc}}$ classes of a single free variable have semimodels:

$$\{12021, 101, 1201, 1021\}$$

. An $\equiv_3$ class realizing $|U|$-many of these has a semimodel with at most $3 + 3 + |U| \times 5$ elements. This upper bound is almost tight: the smallest semimodel of the theory $le < lf < le < \{12021, 101, 1201, 1021\} < ge < gf < ge$ has 14 elements, while this theorem suggests the semimodel

$$000 + 12021 + 101 + 1201 + 1021 + 000,$$

i.e., we can eliminate 8 of the lower-order elements without affecting the $\equiv_3^{\text{semi}}$ class of the semimodel.

References

[1] R. Amit, S. Shelah *The complete finitely axiomatized theories of order are dense*, Israel J. Math. 23 (1976), pp 200-208.

[2] J. Doner, A. Mostowski, A. Tarski *The elementary theory of well-ordering – a metamathematical study*, Logic Colloquium '77 (Proc. Conf., Wroc law, 1977), pp. 154, Stud. Logic Foundations Math., 96, North- Holland, Amsterdam-New York, 1978.

[3] A. Ehrenfeucht *Decidability of the theory of linear ordering relation*, AMS Notices, 6.3.38:556-38, 1959.

[4] A. Ehrenfeucht *An application of games to the completeness problem for formalized theories*, Fund. Math. 49 1960/1961 129141.

[5] T. Green *Properties of Chain Products and EhrenfeuchtFraïssé Games on Chains*, MSci Thesis University of Saskatchewan, 2002.

[6] C. Karp *Finite-Quantifier Equivalence*, Symposium on the Theory of Models, Berkeley; Published, North-Holland Publishing Co. Amsterdam, 1965 pp. 407-412, 1963.

[7] H. Läuchli, J. Leonard *On the elementary theory of linear order*, Fund. Math LIX:109-116, 1966.

[8] E. Lopez-Escobar *Well-orderings and finite quantifiers*, J. Math Society Japan, 20: 477-489, 1968.

[9] A. Meyer *The inherent computational complexity of theories of ordered sets*, Proceedings of the International Congress of Mathematicians (Vancouver, B. C., 1974), Vol. 2, pp. 477482. Canad. Math. Congress, Montreal, Que., 1975.

[10] A. Slomson *Generalized Quantifiers and Well Orderings*, Archiv Math Logik 15:57-73, 1972.

[11] E. Sonenberg *On the Elementary Theory of Inductive Order*, Archiv Math Logik 19:13-22, 1978.

[12] A. Tarski *Grundzüge des Systemenkalküls Zweiter Teil*, Fundamenta Mathematicae 26:283-301, 1936.

4. DYNAMIC EHRENFEUCHT-FRAISSE GAMES ON LINEAR ORDERS IN FIRST ORDER AND INFINITARY LOGIC

Abstract

This paper continues the previous chapter of this thesis. Here we carry out the enumeration described in that paper, for some small values. We solve the Dynamic Ehrenfeucht-Fraïssé Game for linear orders and in strong logics – logics in which the semantics is an Ehrenfeucht-Fraïssé Game played whle a clock runs towards 0, but in which the clock is not a finite number. We write a normal form for formulas of $L_{\omega_1\omega}$ over ordinals, we enumerate the first $\equiv_k$ classes of linear orders, we discuss a linear order with undecidable theory, in which there is uniform decidability of its Σ_n theory for any n, and we go about extending our normal form, or criterion for equivalence, to nonwellfounded logics.

Transitive sets described by descending sequences

This section is intended as an introduction – it indicates how we enumerate a model class by enumerating its local types, and why it is important to compute transitive sets. Topological spaces are inherently local, so their enumeration via local types is simpler than in the model class of linear orers.

Definition 4.0.2: Let $E_0 = \{e\}$. Let E_1 be the power set of E_0. Let f_1 be the function with domain E_1 and range E_0. For each $k > 0$,

Transitivity: we call $U \subseteq E_k$ f_k-*transitive* just in case: whenever $\phi \in \psi$ and $\psi \in U$, there is some $\xi \in U$ such that $f_k(\xi) = \phi$.

The next level: Let E_{k+1} be the set of f_k-transitive subsets of E_k. Define f_{k+1} to have domain E_{k+1} and value $f(U) = \{f(\phi) : \phi \in U\}$.

For instance, f_2 maps $\{\emptyset, \{e\}\}$ and $\{\emptyset\}$ and $\{\{e\}\}$ to $\{e\}$, and maps $\emptyset$ to $\emptyset$. The elements $U \in E_k$ can be viewed as descriptions of transitive sets in a semantics in which k-many variables $x_0 \dots x_{k-1}$ are assigned, in turn – x_0 to an element of U, x_1 to an element of x_0, and so on. If player I aims to show that transitive sets U and V with atom e are different elements of E_k, player I can assign x_0 to an element of U or y_0 to an element of V; player II assigns the other, and the game continues, with player I trying to show that x_0 and y_0 are different elements of E_{k-1}. Player I wins if, for some $i < k$, x_i is empty and y_i is not, or y_i is empty and x_i is not. These are not quantifier-rank-k descriptions of transitive sets, for in the Ehrenfeucht-Fraïssé game on two transitive sets, player I doesn't have to play a descending sequence of elements, $x_{i+1} \in x_i$. For each $k > 0$, f_k is the function which determines the rank-$k-1$ theory of each

rank-k theory of a transitive set, and thus we view E as a language in which transitive sets are described by their descending sequences.

$f_3(U) \in E_2$	is extended by	$U \in E_3$
$\{\emptyset, \{e\}\}$	is extended by	$\{\{\emptyset, \{e\}\}, \{\emptyset\}, \{\{e\}\}, \emptyset\}$
$\{\emptyset, \{e\}\}$	is extended by	$\{\{\emptyset, \{e\}\}, \{\emptyset\}, \emptyset\}$
$\{\emptyset, \{e\}\}$	is extended by	$\{\{\emptyset, \{e\}\}, \{\{e\}\}, \emptyset\}$
$\{\emptyset, \{e\}\}$	is extended by	$\{\{\emptyset, \{e\}\}, \emptyset\}$
$\{\emptyset, \{e\}\}$	is extended by	$\{\{\emptyset\}, \{\{e\}\}, \emptyset\}$
$\{\emptyset, \{e\}\}$	is extended by	$\{\{\emptyset\}, \emptyset\}$
$\{\emptyset, \{e\}\}$	is extended by	$\{\{\{e\}\}, \emptyset\}$
$\{\emptyset\}$	is extended by	$\{\emptyset\}$
$\{\{e\}\}$	is extended by	$\{\{\{e\}\}\}$
$\emptyset$	is extended by	$\emptyset$

By definition 4.0.2, $\{U \in E_k : f(U) = V\} = \{U : (\forall x \in V \exists y \in U f(y) = x) \wedge (\forall y \in U \exists x \in V f(y) = x)\}$, so the size of the first set is

$$\prod_{x \in V} (2^{|\{y \in E_{k-1} : f(y) = x\}|} - 1).$$

Let U_0 contain every element $y \in E_{k-1} : f(y) \neq E_{k-2}$. Because $\{U \in E_k : f(U) = E_{k-1}\}$ contains $\{U \in E_k : f(U) = E_{k-1} \wedge U_0 \subseteq U\}$, the size of the first set is at least the size of the latter, which is $(2^{|\{y \in E_{k-1} : f(y) = E_{k-2}\}|} - 1)$. So, by recursion, $|E_k| > t_k$, where $t_1 = 2$ and $t_{k+1} = 2^{t_k} - 1$. By counting some initial sets more carefully, we can estimate E_k more precisely from below.

Simply because transitive sets are sets, $|E_k| < T_k$, where $T_1 = 2$ and $T_{k+1} = 2^{T_k}$. We can bring this down by counting E_k precisely, and beginning the T-recursion at $|E_k|$. For instance, $|E_4| = 148$, of which $t_4 = 127$ indeed counts the set of $U \in E_4$ such that $f_4(U) = E_2$. We could improve the lower bound t_5 with extra information about f_4:

E_3	$= f_4(U)$ for	127 elements $U \in E_4$
$E_3 \setminus \{\{e\}\}$	$= f_4(U)$ for	7 elements $U \in E_4$
$E_3 \setminus \{\emptyset\}$	$= f_4(U)$ for	7 elements $U \in E_4$
$\{\{\emptyset, \{e\}\}, \emptyset\}$	is f_4 of	$\{\{\{\emptyset, \{e\}\}, \emptyset\}, \emptyset\}$
$\{\{\emptyset\}, \{\{e\}\}, \emptyset\}$	is f_4 of	$\{\{\{\{e\}\}\}, \{\emptyset\}, \emptyset\}$
$\{\{\emptyset\}, \emptyset\}$	is f_4 of	$\{\{\emptyset\}, \emptyset\}$
$\{\{\{e\}\}, \emptyset\}$	is f_4 of	$\{\{\{\{e\}\}\}, \emptyset\}$
$\{\{\{e\}\}\}$	is f_4 of	$\{\{\{\{e\}\}\}\}$
$\{\emptyset\}$	is f_4 of	$\{\emptyset\}$
$\emptyset$	is f_4 of	$\emptyset$

Thus, the number of $U \in E_5$ such that $f_5(U) = E_4$ is $(2^{127} - 1) \times (2^7 - 1) \times (2^7 - 1)$. This is close to the trivial upper bound 2^{148} on E_5.

Order-topological spaces

Consider OT, the class of finite disjoint unions of linear orders, with the topology which is the disjoint union of the order topologies on the linear orders. I.e., OT

is the class of first-order structures (X, τ, ϵ) – in which the only atomic formulas are $x \in U$, for $x \in X$, $U \in \tau$ – for which the topological space (X, τ) which can be extended to models $(X = \lambda_0 \cup \ldots, \lambda_{n-1}, <_0 \cdots <_n, \tau)$, where $<_i$ is a linear order on λ_i, and τ is the topology with basis the set of open intervals $\{x \in \lambda_i : a < x < b\}$, for any parameters $a, b \in \lambda_i$.

Definition 4.0.3: For T_0, T_1 in OT and for $x_0 \in T_0$ and $x_1 \in T_1$, we say these order-topological spaces are *locally equivalent*, symbolized $(T_0, x_0) \equiv_k^{\text{loc}} (T_1, x_1)$, if there is some R in OT such that, taking S_i for the disjoint union of R and T_i, $(S_0, x_0) \equiv_k (S_1, x_1)$.

Topology is a convenient class in which to discuss local formulas. The first-order theory of $(\mathbb{Q} + \mathbb{Z}, 0 \in \mathbb{Q}, 0 \in \mathbb{Z}, 1 \in \mathbb{Z})$ does not know that $0 \in \mathbb{Z}$ and $1 \in \mathbb{Z}$ are neighbors in the ordering. What it can say about these three parameters is entirely *local* to them – that $0 \in \mathbb{Q}$ is a limit of limits of limits ..... and that $0 \in \mathbb{Z}$ is isolated. For any natural number k, and for any T in OT, let T_k be the disjoint union of k-many copies of T. $Th_k(T_k)$ is determined by the set of $\equiv_{k-1}^{\text{loc}}$ classes realized in T.

Theorem 4.0.3: The $\equiv_{2\times k}^{\text{loc}}$ classes and $\equiv_{2\times k+1}^{\text{loc}}$ of order-topological spaces correspond one-to-one with sets in E_k, where the correspondence c maps the $\equiv_{2\times k}^{\text{loc}}$ class of an element which is a limit of the $\equiv_{2\times k-2}^{\text{loc}}$ classes U to the element $\{c(V) : V \in U\}$ of E_k.

Proof:The set U of $\equiv_{2\times k-2}^{\text{loc}}$ classes which limit to x must be transitive, for if elements of type $\phi(y) \in U$ limit to x, and the formula $\phi(y)$ requires that elements of type ψ limit to y, then elements of type ψ limit to x, and a full $\equiv_{2\times k-2}^{\text{loc}}$ description of those elements is part of the $\equiv_{2\times k+1}^{\text{loc}}$ class of (T_i, x_i). □

Furthermore, a set U of $\equiv_k^{\text{loc}}$ classes is the set of $\equiv_k^{\text{loc}}$ classes realized in some order-topological space just in case U is transitive.

Thus, the first-order classes of topological spaces derived from linear orders can be enumerated with arbitrary accuracy. E.g., the number of $\equiv_1 0$ classes of topological spaces T_{10} (which contains 10 copies of T, which prevents the logic from counting the number of elements of some type) is the number of transitive subsets of E_4. Among those transitive subsets are $(2^{127} - 1) \times (2^7 - 1) \times (2^7 - 1)$ subsets of E_4 whose image under f_5 is all of E_4 (i.e., they are transitive by virtue of being large enough); on the other hand, $|E_4| = 148$.

When we study the $\equiv_k$ classes of linear orders, we have an extra complication: that each different permutation of the order in which the various $\equiv_k^{\text{loc}}$ classes have their least element is noticed by Th_{k+2} of the linear order, and likewise the order in which the various $\equiv_k^{\text{loc}}$ classes have their greatest element. Thus, the enumeration of $\equiv_k$ classes of linear orders requires

- the enumeration of transitive sets of local types which can limit to any element, and
- the enumeration of permutations on the ordering of the local types as they approach an element which maintain transitivity in each interval between the appearance or disapperance of one type and the appearance or disapperance of another.

The permutations expand the $\equiv_k$ classes of linear orders, relative to the $\equiv_k$ classes of their induced topologies: there are $\exp_2^k(c_k)$-many $\equiv_k$ classes of linear order (where c_k tends to 1.23 with the first 6 values of k), and there are $\exp_2^{\lfloor (k+1)/2 \rfloor}(c_{\lfloor (k+1)/2 \rfloor})$-many $\equiv_k$ classes of order-topological spaces where $c = (1, 1, 1, 0.8, 0.6, 0.59)$ – i.e., the exponential function is iterated twice as often when computing $\equiv_k$ classes of linear orders as when computing $\equiv_k$ classes of order-topological spaces.

4.1 $\equiv_2$ and $\equiv_3$ classes of linear orders

Theorem 4.1.1: Writing 1 for $Th_1(1)$ and 0 for $Th_1(\emptyset)$, the following sets of pairs of $\equiv_1$ classes of linear orders are consistent:

$$\{(1,1)\}, \{(1,1),(1,0)\}, \{(0,1),(1,1)\}, \{(0,1),(1,1),(1,0)\}, \{(1,0),(0,1)\}, \{(0,0)\}, \emptyset,$$

and the following sets of pairs of $\equiv_1$ classes of linear orders are inconsistent:

$$\{(0,1)\}, \{(1,0)\}, \{(0,0)\} \cup U, \text{ if } U \not\subseteq \{(0,0)\}.$$

Let $W_Z = \{\{(1,1)\}, \{(1,1),(1,0)\}, \{(0,1),(1,1)\}, \{(0,1),(1,1),(1,0)\}, \emptyset\}$. The function ξ is uniformly 1 on W_Z, except that $\xi(\emptyset) = 0$.

- To $(1,1) \in \{(1,1)\}$, assign the pair $(\{(1,1),(1,0)\}, \{(0,1),(1,1)\})$. Writing 1 for $\xi(\{(1,0),(1,1)\})$, $\xi(\{(0,0)\})$, and $\xi(\{(0,1),(1,1)\})$ and applying the definition of addition for sets of pairs of $\equiv_1$ classes, we find: $\{(1,1),(1,0)\} + \{(0,0)\} + \{(0,1),(1,1)\} = \{(1,1+1+1),(1,0+1+1),(1+0,0+1),(1+1+0,1),(1+1+1,1)\} = \{(1,1)\}$ and the pair of ξ values $(\xi(\{(1,1),(1,0)\}), \xi(\{(0,1),(1,1)\})) = (1,1)$ is the chosen element.
- To $(1,1) \in \{(1,1),(1,0)\} \in W_Z$ we assign the pair in W_Z: $(\{(1,1),(1,0)\}, \{(0,1),(1,1),(1,0)\})$. The value of ξ on either element of that pair is 1, so the pair of values of ξ is $(\xi(\{(1,1),(1,0)\}), \xi(\{(0,1),(1,1),(1,0)\})) = (1,1)$ is the chosen element, and $\{(1,1),(1,0)\} + \{(0,0)\} + \{(0,1),(1,1),(1,0)\} = \{(1,1),(1,0)\}$ is the chosen set.
- To $(1,0) \in \{(1,1),(1,0)\}$ assign the pair $(\{(1,1),(1,0)\}, \emptyset)$. The value of ξ on this pair is $(1,0)$, the chosen element. The sets of pairs sum to: $\{(1,1),(1,0)\} + \{(0,0)\}$. On those summands, ξ has the value 1, so $= \{(1,1+1),(1,0+1),(1+0,0)\} = \{(1,1),(1,0)\}$ is the chosen set.
- The choices in $\{(0,1),(1,1)\}$ are symmetric.
- To $(0,1) \in \{(0,1),(1,1),(1,0)\}$, assign the pair $(\emptyset, \{(0,1),(1,1),(1,0)\})$. The value of ξ on this pair is $(0,1)$, the chosen element. The sets of pairs sum to: $\{(0,0)\} + \{(0,1),(1,1),(1,0)\} = \{(0,0+1),(1+0,1),(1+1,1),(1+1,0)\}$ which is the chosen set.
- To $(1,0) \in \{(0,1),(1,1),(1,0)\}$ we assign the pair $(\{(0,1),(1,1),(1,0)\}, \emptyset)$, symmetric to the previous item.

- To $(1,1) \in \{(0,1),(1,1),(1,0)\}$ we assign the pair $(\{(0,1),(1,1),(1,0)\}, \{(0,1),(1,1),(1,0)\})$. Since $\xi(\{(0,1),(1,1),(1,0)\}) = 1$, the pair of ξ values is $(1,1)$, the chosen element. Further, the the sum $\{(0,1),(1,1),(1,0)\}$ $+\{(0,0)\} + \{(0,1),(1,1),(1,0)\} = \{(0,1),(1,1),(1,0)\}$ is the chosen set.

If that strategy is played against an exhaustive strategy of player I, with initial set $\{(1,1)\}$, the set $Z \times \eta$ is constructed: Every time player I plays $(1,1) \in \{(0,1),(1,1),(1,0)\}$, we begin a new copy of Z. Every time player I plays $(1,0)$ or $(0,1)$, we add the immediate predecessor or successor to the greatest or least element of a copy of Z which is already being built. With the other elements of W_Z as initial sets, we get $Z \times \lambda$ for $\lambda = \eta+1, 1+\eta$, or $1+\eta+1$.

Next, let $W_2 = \{\{(1,0),(0,1)\}, \{(0,0)\}, \emptyset, \}$. Again, the value of ξ are 1, except $\xi(\emptyset) = 0$.

- To $(1,0) \in \{(1,0),(0,1)\}$ we assign the pair $(\{(0,0)\}, \emptyset)$. The pair of ξ values are the chosen element. The sum is: $\{(0,0)\}$ $+\{(0,0)\} = \{(0,0+1),(1+0,0)\} = \{(1,0),(0,1)\}$, the chosen set.

- To $(0,1) \in \{(1,0),(0,1)\}$ we assign the pair $(\emptyset, \{(0,0)\})$. The pair of ξ values are the chosen element. The sum is: $\{(0,0)\}$ $+\{(0,0)\} = \{(0,0+1),(1+0,0)\} = \{(1,0),(0,1)\}$, the chosen set.

- To $(0,0) \in \{(0,0)\}$ we assign the pair $(\emptyset, \emptyset)$. The pair of ξ values are the chosen element. The sum is $\emptyset + \{(0,0)\} + \emptyset = \{(0+0,0+0)\} = \{(0,0)\}$, the chosen set.

If that strategy is played against an exhaustive strategy of player I, with initial set $\{(1,0),(0,1)\}$, then we construct the model 2. If we play with initial condition $\{(0,0)\}$, then we construct the model 1.

To see that a set is inconsistent, we use first-order logic.

- Suppose $\{(1,0)\} = \{(\phi,\psi) : \lambda \models \exists x(\phi^{<x} \wedge \psi^{>x})\}$. Since 1, or $Th_1(1)$, is defined by $\exists y(y = y)$, we have: $\lambda \models \exists x(\exists y(y < x) \wedge \neg(\exists y(y > x)))$. But since $\{(1,0)\}$ is a singleton, we also have: $\lambda \models \forall x(\exists y((y < x) \wedge (y = y)) \wedge \neg(\exists y((y > x) \wedge (y = y)))))$. Assign the variables of the first conjunct to $a \in \lambda, b \in \lambda$ so that $b < a$. Now the second conjunct does not hold if we assign x to b. That is: $\{(1,0)\}$ requires that some element is maximal and not minimal, and that every element is maximal and not minimal. But if a is not minimal, there exists $b < a$, and then b is not maximal.

- We treat $\{(0,1)\}$ symmetrically.

- $\{(0,0)\} \cup U$ where $U \not\subseteq \{(0,0)\}$. The element of $U \setminus \{(0,0)\}$ implies $\exists x(\exists y(y < x))$ or $\exists x(\exists y(y > x))$. But $(0,0)$ implies $\exists x(\neg\exists y(y < x) \wedge \neg\exists y(y > x))$. If the former is satisfied with x, y assigned to a, b and the latter is satisfied with x assigned to c, then by totality, a and b are related to c, violating the second formula. □

Applying a similar analysis to $\equiv_3$ is not profitable. Recall that different $\equiv_3$ classes are separated by their labels $I_{\{0,1\}}(\lambda)$ or by the $\equiv_2^{\text{loc}}$ classes at labels and between labels, and the lemma that

$$I_k(\lambda) = I_k^{\text{left}}(\lambda) \cup I_k^{\text{right}}(\lambda).$$

Theorem 4.1.2: Labeling the $\equiv_0^{\mathrm{loc}}$ class e_0, and the $\equiv_1^{\mathrm{loc}}$ class e_1, if the linear order λ has at least five elements, then $I^{\mathrm{left}}_{\{0,1\}}(\lambda)$ is one of the following:

$$(de_0 \in (\emptyset,\emptyset)) = (de_1 \in ((de_0 \in (\emptyset,\emptyset)),\dots)) = (de_0 \in ((de_1 \in ((de_0 \in (\emptyset,\emptyset)),\dots)),\dots)),$$

$$(le_0 \in (\emptyset,\emptyset)) < (de_1 \in ((le_0 \in (\emptyset,\emptyset)),\dots)) = (de_0 \in ((de_1 \in ((le_0 \in (\emptyset,\emptyset)),\dots)),\dots)),$$

$$(le_0 \in (\emptyset,\emptyset)) < (le_1 \in ((le_0 \in (\emptyset,\emptyset)),\dots)) < (de_0 \in ((le_1 \in ((le_0 \in (\emptyset,\emptyset)),\dots)),\dots)),$$

$$(le_0 \in (\emptyset,\emptyset)) < (le_1 \in ((le_0 \in (\emptyset,\emptyset)),\dots)) < (le_0 \in ((le_1 \in ((le_0 \in (\emptyset,\emptyset)),\dots)),\dots)).$$

Proof: The label de_0 is assigned to $(\emptyset,\lambda)$ to indicate that λ has no least element. So for any k, there must be some $\equiv_k^{\mathrm{loc}}$ class such that λ has no least element of that type. There is only one $\equiv_1^{\mathrm{loc}}$ class, e_1. So there is no least element of $\equiv_1^{\mathrm{loc}}$ class e_1. Similarly, the label de_1 must be followed by de_0 again. If λ has at least five elements, then the four labels

$$(le_0 \in (\emptyset,\emptyset)) < (le_1 \in ((le_0 \in (\emptyset,\emptyset)),(ge_0 \in (\emptyset,\emptyset)))) <$$

$$(ge_1 \in ((le_0 \in (\emptyset,\emptyset)),(ge_0 \in (\emptyset,\emptyset)))) < (ge_0 \in (\emptyset,\emptyset))$$

don't exhaust λ. □

Definition 4.1.1: If τ_0 and τ_1 are $\equiv_{k-1}^{\mathrm{loc}}$ classes and if U is a set of $\equiv_{k-1}^{\mathrm{loc}}$ classes, then we call the triple (τ_0, U, τ_1) consistent just in case there is some linear order λ with elements a and b such that (λ, a) has $\equiv_{k-1}^{\mathrm{loc}}$ class τ_0 and (λ, b) has $\equiv_{k-1}^{\mathrm{loc}}$ class τ_1 and the set of $\equiv_{k-1}^{\mathrm{loc}}$ classes realized between a and b is U. If U is a set of $\equiv_{k-1}^{\mathrm{loc}}$ classes and τ_0 is an $\equiv_{k-1}^{\mathrm{loc}}$ class, then we call the triple $(\tau_0, U, -)$ consistent just in case there is some linear order λ and element $a \in \lambda$ such that (λ, a) has $\equiv_{k-1}^{\mathrm{loc}}$ class τ_0 and the set of $\equiv_{k-1}^{\mathrm{loc}}$ classes realized to the right of a is U. Similarly, we define consistency of $(-, U, \tau_1)$. $(-, U, -)$ is consistent just in case it is the set of $\equiv_{k-1}^{\mathrm{loc}}$ classes realized in some linear order.

Theorem 4.1.3: A triple as defined above is consistent just in case there is a set W of such triples, so that for any $(\tau_0, U, \tau_1) \in W$, the following holds:

- For any label described by τ_0, (or, symmetrically, τ_1) defining the least element of type τ', either 1. there is no element of U extending τ', and either 1a. τ_0 itself extends τ', and the union of what all elements of U imply about the the least element of type τ' is consistent with τ_1 being that least element, or 1b. the union of what τ_0 and all elements of U imply about the least element of type τ' is extended by τ_1's description of the least element of type τ' to the right of the triple, or 2. there is a pair $((\tau_0, U_0, \tau), (\tau, U_1, \tau_1))$ of triples in W such that τ extends τ' and U_0 contains no element extending τ', and $U_0 \cup \{\tau\} \cup U_1 = U$.

- For any label described by τ_0, (or, symmetrically, τ_1) defining a descending sequence of elements of type τ', either 1a. that sequence limits to τ_0, and $(-, U, \tau_1)$ is consistent and whenever the strategy splits $(-, U, \tau_1)$ into a pair (V_0, V_1) of triples in W, there is an element extending τ' in V_0, or 1b. there is no element of U extending τ', and conditions 1b from the previous item hold, or 2. there is a pair $((\tau_0, U_0, -), (-, U_1, \tau_1))$ of triples in W such that U_0 contains no element extending τ' and $U_0 \cup U_1 = U$, and, further, whenever the strategy splits $(-, U_1, \tau_1)$ into a pair (V_0, V_1) of triples in W, there is an element extending τ' in V_0.
- Symmetric conditions explain how each label described by τ_1 is realized at τ_0 or is realized below τ_0 or can be realized within U, splitting the triple into two triples, to be realized to its right and its left.
- For any element $\tau \in U$, there is a pair $((\tau_0, U_0, \tau), (\tau, U_1, \tau_1))$ of elements of W such that $U_0 \cup \{\tau\} \cup U_1 = U$.

Proof: Given W, player II can last arbitrarily long in the linear consistency game. If player I plays an exhaustive strategy, then the result will be a linear order λ of elements between τ_0 and τ_1 in which τ_0 implies $Th^{\text{left}}_{k-1}(\lambda)$, τ_1 implies $Th^{\text{right}}_{k-1}(\lambda)$, and U is the set of $\equiv^{\text{loc}}_{k-1}$ classes realized. If τ_1 exists, then the conditions in the first two items imply that the description of the linear order right of the triple given by τ_1 extends what all other played constants imply about any label, we can choose $\{c \in \lambda : c > b\}$ to be any element of the $\equiv^{\text{left}}$ class that is the right part of τ_1; we can choose likewise $\{c \in \lambda : c < a\}$ and we have the linear order with constants (λ, a, b) that proves the triple to be consistent. □

Now we apply the theorem to enumerate $\equiv_3$ classes of linear orders. In $I^{\text{left}}_{\{0,1\}}(\lambda)$, all of the labels either 1. label the same gap as the preceding label, or 2. label the element of λ which is the immediate successor of the preceding label. If $\lambda = 5$, then $I^{\text{left}}_{\{0,1\}}(\lambda)$ and $I^{\text{right}}_{\{0,1\}}(\lambda)$ overlap. If $|\lambda| > 5$, they don't.

This simplifies using the local linear consistency game: we hypothesize different combinations of $I^{\text{left}}_{\{0,1\}}(\lambda)$ and $I^{\text{right}}_{\{0,1\}}(\lambda)$, hypothesize different $\equiv^{\text{loc}}_2$ classes for the greatest element of $I^{\text{left}}_{\{0,1\}}(\lambda)$ and the least element of $I^{\text{left}}_{\{0,1\}}(\lambda)$, hypothesize a set of $\equiv^{\text{loc}}_2$ classes to be realized between them, and the determine the resulting set to be consistent (using the preceding theorem) or inconsistent (using first-order logic).

Theorem 4.1.4: There are four inextensible $\equiv^{\text{loc}}_2$ classes.

Proof: $I^{\text{left}}_{\{0\}}$ describes nothing (and is extensible) or describes either le or de, and nothing more. $I^{\text{right}}_{\{0\}}$ likewise describes nothing (and is extensible) or describes ge or ae. The four formulas $ge^{>x} \wedge le^{<lx}$, $ge^{>x} \wedge de^{<lx}$, $ae^{>x} \wedge le^{<lx}$, $ae^{>x} \wedge de^{<lx}$ are $\not\equiv^{\text{loc}}_2$ and exhaust the inextensible $\equiv^{\text{loc}}_2$ classes. □

We abbreviate the $\equiv^{\text{loc}}_2$ classes in U as $(ge, le), (ge, de), (ae, le), (ae, de)$ and as gl, gd, al, ad. We abbreviate the $\equiv^{\text{loc}}_2$ class of the left element τ_0 as d or l (since there's no reason examining $\tau_0 = ad$ and $\tau_0 = gd$ independently) and we allow d to stand for d or $-$, the absence of any τ_0, since either one is consistent just in case the other is. We abbreviate the $\equiv^{\text{loc}}_2$ class on the right as a or g and let a represent both a and $-$, likewise. We proceed to consider which triples are consistent:

Theorem 4.1.5: The following lists all triples and which are consistent and inconsistent:

1. $(l, \emptyset, g)$ is consistent.
2. $(l, \emptyset, a)$, $(d, \emptyset, g)$, and $(d, \emptyset, a)$ are inconsistent.
3. $(d, \{ad\}, a)$ is consistent.
4. $(l, \{ad\}, g)$, $(l, \{ad\}, a)$, and $(d, \{ad\}, g)$ are inconsistent.
5. $(l, \{ad, gd\}, a)$ is consistent.
6. $(l, \{ad, gd\}, g)$, $(d, \{ad, gd\}, g)$, and $(d, \{ad, gd\}, a)$ are inconsistent.
7. $(d, \{ad, al\}, g)$ is consistent.
8. $(l, \{ad, al\}, g)$, $(l, \{ad, al\}, a)$, and $(d, \{ad, al\}, a)$ are consistent.
9. Any triple with $U = \{al, gd\}$, $U = \{al, gd, ad\}$, or U containing $\{gl\}$ is consistent.
10. Any triple with $U = \{al\}$ or $U = \{gd\}$ is inconsistent.

Proof:

1. In the local linear consistency game, we only have to explain where l, the least element above the left end, is realized and where g, the greatest element below the right end, is realized. l is realized at τ_1 and g is realized at τ_0. Both of these satisfy condition 1a. in the theorem on the local linear consistency game.
2. If there is no element of λ between two cuts, those cuts are the same cut. We can't have different cuts separated by $\emptyset$. Consider, for instance, that the left end of the cut is $-$ or d, either because $I^{\text{left}}_{\{0,1\}}(\lambda)$'s last element is $de\ldots$, or because $I^{\text{left}}_{\{0,1\}}(\lambda) = le \in (\emptyset, \emptyset) < lf \in (le, \ldots) < le \in (lf, \ldots)$, and that last element has $\equiv^{\text{loc}}_2$ class (ge, de). These situations require an infinite descending sequence of elements, which is not supplied by $U = \emptyset$.
3. The label d describes a sequence which limits to τ_0; likewise, the label a describes a sequence which limits to τ_1. For the unique element of U, we split U into $(-, U, a), (d, U, -)$.
4. l describes the immediate successor to τ_0. If we are to realize both τ_0 and l in a linear order, then l must have an $\equiv^{\text{loc}}_2$ class consistent with its having an immediate predecessor.
5. For l, the immediate successor of τ_0, we choose $gd \in U$ and split U into the pair: $(l, \emptyset, gd), (gd, \{ad\}, a)$. Those are found to be consistent in previous items.
6. g describes the immediate predecessor to τ_1. So it must have an $\equiv^{\text{loc}}_2$ class consistent with having an immediate successor. To $gd \in U$ corresponds a cut, in which U is split into $(?, U_0, gd)$ and $(gd, U_1, ?)$. The g in the first of these triples requires an immediate predecessor, so either U_0 contains

some $\equiv_2^{\mathrm{loc}}$ class consistent with having an immediate successor (which doesn't exist in $U = \{gd, ad\}$) or $U_0 = \emptyset$. But $(d, \emptyset, gd)$ is inconsistent, by a previous item.

7. symmetric to item 5.

8. symmetric to item 6.

9. Let $U = \{al, gd\}$. In $(l, U, ?)$, we assign to l the pair $(l, \emptyset, gd), (gd, U, ?)$. Symmetrically, in $(?, U, g)$, we assign to g the pair $(?, U, al), (al, \emptyset, g)$. In $(-, U, -)$, to $gd \in U$ we assign the pair $(-, U, gd), (gd, U, -)$, and to $al \in U$ we assign the pair $(-, U, al), (al, U, -)$. Now the five triples: $\{(l, U, g), (l, U, a), (d, U, g), (d, U, a), (l, \emptyset, g)\}$ are all consistent, since we can pass from any information in any of the first four to a pair of them, satisfying consistency conditions, and the fifth is consistent. We can split $(?, \{al, gd, ad\}, ?)$ on ad into $(?, \{al, gd\}, ad), (ad, \{al, gd\}, ?)$, which we have just found to be consistent. If $gl \in U$, we have a simple procedure: answer any l in $(l, U, ?)$ with $(l, \emptyset, gl)(gl, U, ?)$; answer any g in $(?, U, g)$ with $(?, U, gl), (gl, \emptyset, g)$; split U always into $U_0 = U_1 = U$.

10. Realize $al \in \{al\}$ at x_0. x_0 requires a sequence of predecessors in with $\equiv_2^{\mathrm{loc}}$ class in $\{al\}$. Realize one such at x_1. x_1 requires an immediate successor. Let x_2 be that successor. Now the $\equiv_2^{\mathrm{loc}}$ class of (λ, x_2) recognizes that x_2 has an immediate predecessor, so it is not al. The case of $U = \{gd\}$ is symmetric. □

With that theorem it is easy to enumerate the $\equiv_3$ classes of linear orders:

- 1. The linear order 6, in which $I^{\mathrm{left}}_{\{0,1\}}$ assigns its maximal element to $2 \in 6$ and $I^{\mathrm{right}}_{\{0,1\}}$ assigns its minimal element to $3 \in 6$, and no element of 6 is realized between them: 1.
- 3. There are four ways $I^{\mathrm{left}}_{\{0,1\}}$ could assign its maximal element to a cut, or assign its maximal element into λ, to an element with $\equiv_2^{\mathrm{loc}}$ class (ge, de). Likewise, there are four ways $Th^{\mathrm{right}}(\lambda)$ could require an ascending sequence (a), not a maximal element (g): +16.
- 5. Only one $I^{\mathrm{left}}_{\{0,1\}}$ assigns its maximal element into λ, to an element of $\equiv_2^{\mathrm{loc}}$ class (ge, le). There are four ways $I^{\mathrm{right}}_{\{0,1\}}$ could assign its minimal element to a cut, or assign its minimal element into λ, to an element with $\equiv_2^{\mathrm{loc}}$ class (ae, le): +4.
- 7. Symmetric to the previous item: +4.
- 9. These 10 sets U are consistent with all 5 possible $Th^{\mathrm{left}}(\lambda)$ classes and all 5 possible $Th^{\mathrm{right}}(\lambda)$ classes: $+10 \times 25$.
- There are five linear orders of size ≤ 4: +5.
- In the linear order 5, the greatest element of $I^{\mathrm{left}}_{\{0,1\}}$ and the least element of $I^{\mathrm{right}}_{\{0,1\}}$ are assigned to the same element $3 \in 5$: +1.

It was not profitable to form $\equiv_2^{\mathrm{loc}}$-almost locally sets, because the $\equiv_2^{\mathrm{loc}}$ classes are very independent. To enumerate $\equiv_4$, we will surely need to know $\equiv_3^{\mathrm{loc}}$-almost locally sets.

4.2 Enumerating $\equiv_4$ classes of linear order

In this section we generate trees of labels that might be $I_{\{2,1,0\}}(\lambda)$ for some linear order λ and we will hypothesize $\equiv_3^{\text{loc}}$ classes that might complete $Th_4(\lambda)$. We'll use the local consistency game to find which of these are in fact consistent. The result is an effective algorithm for enumerating $\equiv_4$ classes of linear orders. We will write e for the unique $\equiv_0^{\text{loc}}$ class and f for the unique $\equiv_1^{\text{loc}}$ class. We first consider "short" assignments $I_{\{1,0\}}$, and then we consider assignments $I_{\{1,0\}} = I^{\text{left}}_{\{1,0\}} + I^{\text{right}}_{\{1,0\}}$ (by $+$ we refer to the union of $I^{\text{left}}_{\{1,0\}}$ and $I^{\text{right}}_{\{1,0\}}$, with every element of $I^{\text{left}}_{\{1,0\}}$ preceding every element of $I^{\text{right}}_{\{1,0\}}$). There are many different $\equiv_2^{\text{loc}}$ refinements of $I^{\text{left}}_{\{1,0\}} + I^{\text{right}}_{\{1,0\}}$ because, unlike $\equiv_0^{\text{loc}}$ and $\equiv_1^{\text{loc}}$, $\equiv_2^{\text{loc}}$ is not a singleton. It has four inextensible elements and the intervals in which the $\equiv_2^{\text{loc}}$ classes exist can overlap in various ways. We refine each possible assignment $I_{\{2\}}$ with three singleton equivalence relations, in turn, and the results are all the possible assignments $I_{\{2,1,0\}}$. Then we assign an $\equiv_3^{\text{loc}}$ class to each element of $I_{\{2,1,0\}}$ and we assign sets of $\equiv_3^{\text{loc}}$ classes to each gap in $I_{\{2,1,0\}}$.

Definition 4.2.1: This is our algorithm for enumerating $\equiv_4$ classes of linear orders:

1. Enumerate assignments $I_{\{0\}} = I_\emptyset$, $I_{\{1\}} = I_{\{0\}}$, and $I_{\{1,0\}} = I_{\{1\}}$ (where the last refinement adds no labels) and assignments $I_{\{0\}}$, $I_{\{1\}}$, and $I_{\{1,0\}}$ in which the last label in I^{left}_τ and some right label in I^{right}_τ are assigned to the same element.[1] Hereafter, assume that the first three refinements of $I_\emptyset$ are nontrivial and that $I_{\{1,0\}} = I^{\text{left}}_{\{1,0\}} + I^{\text{right}}_{\{1,0\}}$.

2. Start with the *Stack* $= \{I^{\text{left}}_\emptyset\} = \{\emptyset\}$.

3. For each sequence A in the *Stack*, refine $(A, \emptyset)$ with respect to the singleton equivalence relation $\equiv_0^{\text{loc}}$ and append to A any labels defined in the refinement which are constant across an $\equiv_4^{\text{left}}$ class. Put the result in the *Stack*.

4. For each sequence A in the *Stack*, refine the left half of the final cut, $(A, \emptyset)$ with respect to the singleton equivalence relation $\equiv_1^{\text{loc}}$ and put the list of all possible refinements in the *Stack*.[2]

5. For each sequence A in the *Stack*, refine the left half of the final cut, $(A, \emptyset)$ with respect to the singleton equivalence relation $\equiv_0^{\text{loc}}$ and put the list of all possible refinements in the *Stack*.[3]

6. Remove each sequence A from the *Stack* and do the following:

[1] There are only five such assignments, and they are enumerated in the proof.

[2] If the last label in A is $'de'$, the only left refinement of $(A, \emptyset)$ is A with a new label $'df'$ assigned to be equal to the last element of A. Otherwise, return the two assignments: A with the new greatest element $'lf'$, and A with the new greatest element $'df'$.

[3] If the last label in A is $'df'$, the only left refinement of $(A, \emptyset)$ is A with a new label $'de'$ assigned to be equal to the last element of A. Otherwise, return the two assignments: A with the new greatest element $'le'$, and A with the new greatest element $'de'$.

(a) For each label m which could follow A, append m to A and put the result back onto the *Stack*.[4]
(b) if A doesn't end $'de-'$ or $'df-'$, add A to the list of $I^{\text{left}}_{\{2\}}$ assignments.

7. Group $I^{\text{left}}_{\{2\}}$ assignments into a common class if they agree on $I^{\text{left}}_{\{2\}} \setminus I^{\text{left}}_{\{1,0\}}$.[5]
8. Group $I^{\text{left}}_{\{2\}}$ assignments by the set of $\equiv^{\text{loc}}_2$ classes which are labeled.
9. Generate and group $I^{\text{right}}_{\{2\}}$ assignments, repeating steps 2 through 8.
10. For each $I^{\text{left}}_{\{2\}}$ and $I^{\text{right}}_{\{2\}}$ which label the same $\equiv^{\text{loc}}_2$ classes, we now determine all orderings of $I^{\text{left}}_{\{2\}} \cup I^{\text{right}}_{\{2\}} = I_{\{2\}}$starting with the *Stack* containing one element – the assignment $I^{\text{left}}_{\{2\}} + I^{\text{right}}_{\{2\}}$. Remove each assignment I from the *Stack*, and do the following:
    (a) Add I to the set of orderings of $I^{\text{left}}_{\{2\}} \cup I^{\text{right}}_{\{2\}}$.
    (b) For each ordered pair of labels $p < q$ in I, if p is "$d\ldots$" or "$l\ldots$" and q is "$a\ldots$" or "$g\ldots$" and p and q don't label the same $\equiv^{\text{loc}}_2$ type, switch them so that $q < p$ and push the result onto the *Stack*.
    (c) For each ordered pair of labels $p < q$ in I, if $p =$ "$l\alpha \in (I^{\text{left}}_{\{1,0\}}, \emptyset)$" and $q =$ "$g\alpha \in (\emptyset, I^{\text{right}}_{\{1,0\}})$", replace p and q by the single label "$o\alpha \in (I^{\text{left}}_{\{1,0\}}, I^{\text{right}}_{\{1,0\}})$"[6] and push the result onto the *Stack*.
11. Part (a) of the previous item has listed the set of possible $I_{\{2\}}$ assignments. For each assignment $I_{\{2\}}$, initialize a set *Available* of $\equiv^{\text{loc}}_2$ classes so that $\mathit{Available} = \emptyset$. Go through the ordered pairs of labels $p < q$ in $I_{\{2\}}$ in turn from least to greatest (from left to right) and:
    (a) if p is "$d\ldots$" or "$l\ldots$" and m labels $\equiv^{\text{loc}}_2$ class α, then add α to *Available*.
    (b) if q is "$a\ldots$" or "$g\ldots$" and m labels $\equiv^{\text{loc}}_2$ class α, then remove α from *Available*.
    (c) look up the triple (p, *Available*, q) in a *Branching Table*, which determines the possible refinements $I_{\{2,0\}}$ $I_{\{2,1\}}$ and $I_{\{2,1,0\}}$ in that cut of $I_{\{2\}}$ and, for each refinement, the possible $\equiv^{\text{loc}}_3$ class of each label and the set of $\equiv^{\text{loc}}_3$ classes realized in each cut.

The *Branching Table* constructs all possible assignments $I_{\{1,0\}}[p,q] = \{$labels $m \in I_{\{2,1,0\}}$ such that $p \leq m \leq q\}$ as the union $I^{\text{left}}_{\{1,0\}}[p,\emptyset) \cup I^{\text{right}}_{\{1,0\}}(\emptyset,q]$ so as to list the possible left and right assignments in the $\equiv^{\text{loc}}_0$ refinement of the $\equiv^{\text{loc}}_1$ refinement of the $\equiv^{\text{loc}}_0$ refinement of (p,q) separately. The $\equiv^{\text{loc}}_0$ and $\equiv^{\text{loc}}_1$ refinements label the least element above p just in case $p \neq {}'a\ldots'$. In that case,

1. If (p,q) could be empty[7] then add that to the set of possible assignments

[4] The subroutine which finds the labels which could follow A is called the routine of *Left Legal Extensions* and is described at the end of this definition.

[5] This is similar to the classification of $\equiv^{\text{left}}_3$ classes into $\{de, le-df, le-lf-de, le-lf-le$ with the last label in $(ge, de)\}$ and $\{le-lf-le$ with the last label in $(ge, le)\}$ which was useful when enumerating $\equiv_3$ classes.

[6] This labels the "only" element of type α in the cut $(I^{\text{left}}_{\{1,0\}}, I^{\text{right}}_{\{1,0\}}) \in I^{+}_{\{1,0\}}$.

[7] That is, p could be the predecessor of q, and q could be the successor of p.

$I_{\{1,0\}}[p,q]$.

2. If least elements above p and greatest elements below q are both labeled by $\equiv_n^{\text{loc}}$ refinements of $\{p,q\}$, for $n<2$, consider that (p,q) may be exhausted before we refine it three times.

3. If the least elements above p are labeled, and the greatest elements below q are not, consider that (p,q) may be exhausted before we define three successors to p.

4. If the last elements below q are labeled, and the least elements above p are not, consider that (p,q) may be exhausted before we define three predecessors to q.

5. Hereafter, assume that $I_{\{1,0\}}[p,q] = I^{\text{left}}_{\{1,0\}}[p,\emptyset)+I^{\text{right}}_{\{1,0\}}(\emptyset,q]$, where $+$ refers to the ordering on the union in which the left ordered set precedes the right ordered set. We'll list possible assignments $I^{\text{left}}_{\{1,0\}}[p,\emptyset)$ first. If the least element(s) greater than p should be labeled in the $\equiv_n$ refinement of $\{p,q\}$ for $n<2$, start with *Stack* the singleton containing only $I^{\text{left}}_{\emptyset}[p,\emptyset) = \{p\}$. If the least element(s) greater than p should remain unlabeled in a low-rank refinement of $\{p,q\}$, let *Stack* the singleton containing only the word *unlabeled* and skip steps 6 through 9. [8]

6. For each sequence A in the *Stack*, refine the left half of the final cut, $(A,\emptyset)$ with respect to the singleton equivalence relation $\equiv_0^{\text{loc}}$ and put the list of all possible refinements in the *Stack*.[9]

7. For each sequence A in the *Stack*, refine the left half of the final cut, $(A,\emptyset)$ with respect to the singleton equivalence relation $\equiv_1^{\text{loc}}$ and put the list of all possible refinements in the *Stack*.[10]

8. For each sequence A in the *Stack*, refine the left half of the final cut, $(A,\emptyset)$ with respect to the singleton equivalence relation $\equiv_0^{\text{loc}}$ and put the list of all possible refinements in the *Stack*.[11]

---

[8] We will pair this *Stack* to the list of possibilities for $I^{\text{right}}_{\{1,0\}}(\emptyset,q]$ and sets U of $\equiv_3^{\text{loc}}$ types and determine which sets U are consistent with which left and right ends.

[9] The possible refinements of the left half of $(A,\emptyset)$ are:

(a) If the last label in A is $'l\ldots'$ or $'g\ldots'$ or $'o\ldots'$ and labels the $\equiv_2^{\text{loc}}$ type (ge,de) or (ae,de) and *Available* is not $\emptyset$, then the only refinement of the left end of $(A,\emptyset)$ with respect to a singleton equivalence relation is the assignment which assigns the label $'de \in (A,\emptyset)'$ to immediately follow the last element of A (by the natural ordering on elements and cuts, $p<(\{x\in\lambda : x\leq p\},\{x\in\lambda : x>p\})$, and no cut or element is between that element and that cut), and that assignment is a consistent refinement of the left end of $(A,\emptyset)$.

(b) If the last label in A is $'le'$ or $'lf'$ and *Available* contains (ge,de), then the label $'de'$ can follow A.

(c) If the last label in A is $'l\ldots'$ or $'g\ldots'$ or $'o\ldots'$ and labels the $\equiv_2^{\text{loc}}$ type (ge,le) or (ae,le), then the label $'le'$ can follow A.

(d) If the last label in A is $'le'$ and *Available* contains (ge,le), then the label $'le'$ can follow A.

[10] This is the same as the previous item, but with $\equiv_1^{\text{loc}}$ class f replacing $\equiv_0^{\text{loc}}$ class e.

[11] This is the same as footnote 10.

9. For each sequence A in the *Stack*, choose the possible $\equiv_3^{\text{loc}}$ classes of the last element of A.[12]

Symmetrically, construct possible assignments of labels $I^{\text{right}}_{\{1,0\}}(\emptyset, q]$ and pick $\equiv_3^{\text{loc}}$ classes for each label. Then, for each assignment $I^{\text{left}}_{\{1,0\}}[p, \emptyset)$ and $\equiv_3^{\text{loc}}$ classes for those labels, for each assignment $I^{\text{right}}_{\{1,0\}}(\emptyset, q]$ and $\equiv_3^{\text{loc}}$ classes for those labels, construct all possible sets U of $\equiv_3^{\text{loc}}$ types to be realized between ($I^{\text{left}}_{\{1,0\}}[p, \emptyset)$ and $I^{\text{right}}_{\{1,0\}}(\emptyset, q]$). [13] The $\equiv_3^{\text{loc}}$ types are always realized in groups, which describe

1. A sequence of n elements with a sequence of elements descending from above to the greatest of the n elements and a sequence of elements ascending from below to the least of the n elements, for $1 \leq n \leq 7$.

2. The $\equiv_3^{\text{loc}}$ class (l, r) such that l describes 3 elements $ge - gf - ge$, the least of which is in $\equiv_2^{\text{loc}}$ class (ae, le) indicating that it is the limit of a sequence of elements ascending from below, and r describes 3 elements $le - lf - le$, the greatest of which is in $\equiv_2^{\text{loc}}$ class (le, ge) indicating that it has an immediate successor.

3. The $\equiv_3^{\text{loc}}$ class (l, r) such that r describes 3 elements of which the greatest is the limit of a sequence of elements descending from above and l describes 3 elements of which the least has an immediate predecessor.

4. The $\equiv_3^{\text{loc}}$ class (l, r) such that r describes 3 elements, the third of which has an immediate successor and l describes 3 elements, the third of which has an immediate predecessor.

We name these sets of $\equiv_3^{\text{loc}}$ classes:

1. $1 = \{(ae = af = ae, de = df = de)\}$ contains the isolated element by itself.

2. $2 = \{(ae = af - ge, de = df = de), (ae = af = ae, le - df = de)\}$ contains a pair of 2 elements, one the immediate successor of the other, with sequences limiting to them from above and below.

3. $N = \{3, 4, 5, 6, 7\}$, where $3 = \{(ae - gf - ge, de = df = de), (ae = af - ge, le - df = de), (ae = af = ae, le - lf - de)\}$ and in general, n has n elements $(2 < n < 8)$, describes the finite set of between three and seven element elements, with sequences limiting to them from above and below.

4. The last three sets of a single $\equiv_3^{\text{loc}}$ class each, as enumerated above, we name: W, X, and Z.

[12] If the last label of A is $'le\ldots'$ and if *Available* contains (ge, le) then assign the $\equiv_3^{\text{loc}}$ class (l, r) where l is determined by A (that is, l gives the names $ge - gf - ge$ to the elements $p - le - lf$ of A, and l assigns the $\equiv_2^{\text{loc}}$ class to ge or p which is the $\equiv_2^{\text{loc}}$ class which p labels) and r is the $\equiv_3^{\text{left}}$ class with $I^{\text{left}}_{\{1,0\}} = le - lf - le$ and the last element in $\equiv_2^{\text{loc}}$ class (ge, le). If the last label of A is $'le\ldots'$ and if *Available* contains (ge, de) then assign the $\equiv_3^{\text{loc}}$ class (l, r) where l is determined by A, and r is any of the four $\equiv_3^{\text{loc}}$ classes with $I^{\text{left}}_{\{1,0\}} = de, le - df, le - lf - de$, or $le - lf - le$ with the last element in $\equiv_2^{\text{loc}}$ class (ge, de).

[13] That is done by the subroutine $Con(U)$.

The subroutine Con(U) declares the triple $(I^{\text{left}}_{\{1,0\}}[p,\emptyset), U, I^{\text{right}}_{\{1,0\}}(\emptyset,q])$ to be consistent just in case the following tests are met:[14]

1. Each element of U implies the existence of a number of $\equiv^{\text{loc}}_2$ types. Check that all of those are in *Available*.
2. If $I_{\{1,0\}}[p,q]$ contains the label $'de'$ or $'ae'$, then $U \neq \emptyset$.
3. If p is $'dx'_0 = \ 'dx'_1 \ldots$ or q is $'ax'_0 = \ 'ax'_1 \ldots$, check that U contains an extension of each x_i into $\equiv^{\text{loc}}_3$.
4. If U contains W then it must contain X or Z or $I^{\text{right}}_{\{1,0\}}(\emptyset,q]$ has three predecessors to q, the least of which requires a predecessor.
5. If U contains X then it must contain W or Z or $I^{\text{left}}_{\{1,0\}}[p,\emptyset)$ has three successors to p, the last of which requires a successor.
6. If $I^{\text{left}}_{\{1,0\}}[p,\emptyset)$ has three successors to p, the last of which requires a successor, then U contains X or Z or U is empty and $I^{\text{right}}_{\{1,0\}}(\emptyset,q]$ has three predecessors to q, the least of which requires a predecessor.
7. If $I^{\text{right}}_{\{1,0\}}(\emptyset,q]$ has three predecessors to q, the least of which requires a predecessor, then U contains W or Z or U is empty and $I^{\text{left}}_{\{1,0\}}[p,\emptyset)$ has three successors to p, the last of which requires a successor.
8. If $I^{\text{left}}_{\{1,0\}}[p,\emptyset)$ requires a sequence limiting to p or to $p-le$ or to $p-le-lf$ or to $p-le-lf-le$, then U contains 1 or 2 or Z or an element of N or both W and X.
9. If $I^{\text{right}}_{\{1,0\}}(\emptyset,q]$ requires a sequence limiting to q or to $ge-q$ or to $gf-ge-q$ or to $ge-gf-ge-q$, then U contains 1 or 2 or Z or an element of N or both W and X.
10. If $W \in U$ then U contains 1 or 2 or Z or X or an element of N.
11. If $X \in U$ then U contains 1 or 2 or Z or W or an element of N.

[14] Footnote 7 carefully describes the case in which $(p,q) = \emptyset$ and the $\equiv^{\text{loc}}_0$ refinement of $[p,q]$ describes q as the immediate successor of p and describes p as the immediate predecessor of q. This leaves the case in which the $\equiv^{\text{loc}}_0$ refinement of $[p,q]$ doesn't describe the least element(s) above p or doesn't describe the greatest element(s) below q, so that the current subroutine will describe, for instance, $[p-$unknown left $-U = \emptyset-$unknown right$-q]$. Step 2 of the branching table describes $[p-le-df = de = q = (U = \emptyset) =$unknown right$= q]$, since (p,q) is exhausted before three refinements can be made, with $I^{\text{left}}_{\{1,0\}}[p,\emptyset)$ preceding $I^{\text{right}}_{\{1,0\}}(\emptyset,q]$. However, $le-df = de$ is a complete $I^{\text{left}}_{\{1,0\}}[p,\emptyset)$ assignment, so we will consider that triple again in this subroutine. Rather than make arbitrary distinctions, when writing code we double-check the counting of $U = \emptyset$ by listing all short possibilities (for $I_{\{1,0\}}[p,q]$, $\equiv^{\text{loc}}_3$ classes of those labels, and the set $U = \emptyset$ of $\equiv^{\text{loc}}_3$ classes in the center) and then listing those which are accepted by Con(U), and then identifying and removing redundancy.

This is not to blur the theoretical distinction between assignments $I_{\{1,0\}}[p,q]$ which assign the least and greatest (sequences of, or single) elements of $\equiv^{\text{loc}}_0$ type e or declare that element to be unknown, and those which know the least and greatest elements but find the interval to be empty – either the $\equiv^{\text{loc}}_0$ refinement of $I_\emptyset(\{p,q\}) = \{p,q\}$, $I_{\{0\}}[p,q]$, or the next refinement, $I_{\{1\}}[p,q]$, or the next, $I_{\{1,0\}}[p,q]$, assign the least and greatest element to be equal, and be the unique element, or assign no labels at all.

The subroutine *Left Legal Extensions* accepts an assignment A and determines which labels might follow A as parts of an $\equiv_2^{\text{loc}}$ refinement of $I_{\{1,0\}}$.

1. initialize a set *Available*
2. If A labels the $\equiv_2^{\text{loc}}$ class (ae, le) or (ge, le), add the term *predecessor* to *Available*.
3. If A labels the $\equiv_2^{\text{loc}}$ class (ge, de) or (ge, le), add the term *successor* to *Available*.
4. If A labels the $\equiv_2^{\text{loc}}$ class (ge, le) or (ae, de) or both (ae, le) and (ge, de), add the term *repeatable* to *Available*.
5. if the last label in A is $'le'$ or $'lx'$ for x one of the $\equiv_2^{\text{loc}}$ classes (ae, le) or (ge, le),
   (a) if *Available* contains the term *successor*,
      i. if *Available* contains the term *predecessor*, then $'lx'$ can follow A, for x either (ge, de) or (ge, le).
      ii. $'lx'$ can follow A, for x either (ae, le) or (ae, de).
      iii. A can be followed by $dx_0 = dx_1 = \ldots$, for $\{x_i : i \leq n\}$ any consistent descending sequence.[15]
   (b) if *Available* doesn't contain the term *successor*, then only $'lx'$ can follow A, for x either (ge, de) or (ge, le).
6. if the last label in A is $'le'$ or $'lx'$ (for x one of the $\equiv_2^{\text{loc}}$ classes (ae, de) or (ge, de)) or $'dx'$ for any $\equiv_2^{\text{loc}}$ class x,
   (a) if *Available* contains the term *repeatable*,
      i. if *Available* contains the term *predecessor*, then $'lx'$ can follow A, for x either (ge, de) or (ge, le).
      ii. $'lx'$ can follow A, for x either (ae, le) or (ae, de).
   (b) A can be followed by $dx_0 = dx_1 = \ldots$, for $\{x_i : i \leq n\}$ any consistent descending sequence.[16]

Theorem 4.2.1: The program in the preceding definition enumerates $\equiv_4$ classes of linear orders.

Proof:

- Steps 1 through 5 list all possible assignments $I_{\{1,0\}}$. Step 1. lists those assignments $I_{\{1,0\}}$ for which it does not hold that $I_{\{1,0\}}^{\text{left}} < I_{\{1,0\}}^{\text{right}}$. Those are $I_{\{0\}} = \emptyset, I_{\{0\}} = oe, I_{\{1\}} = le - \emptyset - ge, I_{\{1\}} = le - of - ge, I_{\{1,0\}} = le - lf - \emptyset - gf - ge, I_{\{1,0\}} = le - lf - oe - gf - ge$. This same list

[15] For each nonempty set U of $\equiv_2^{\text{loc}}$ types, if

A. $((ae, le) \in U \rightarrow ((ge, le) \in U$ or $(ge, de) \in U$ or *Available* contains the term *predecessor*)) and

B. $((ge, ae) \in U \rightarrow ((ge, le) \in U$ or $(ae, le) \in U$ or *Available* contains the term *successor*)),

then the labels $\{'dx' : x \in U\}$ can all be equal, and be the next distinct element of A.

[16] See the previous footnote.

was constructed as an initial step in enumerating $\equiv_3$ theories of linear order. Steps 2 through 5 define $I^{\text{left}}_{\{1,0\}}$: Step 2 initializes $I^{\text{left}}_{\emptyset} = \emptyset$. Steps 3 through 5 add $'de'$ or $'le'$, $'df'$ or $'lf'$, and $'de'$ or $'le'$. When enumerating $\equiv_3$ theories of linear order, we found that there are four possibilities for $I^{\text{left}}_{\{1,0\}}$. But at this point our program ceases to imitate the enumeration of $\equiv_3$ classes of linear orders.

- Step 6 defines the $\equiv^{\text{left}}_4$ equivalence class of the $\equiv^{\text{loc}}_2$ refinement of $I^{\text{left}}_{\{1,0\}}$ by adding a label to A, then putting the result back on the stack. Step 6 calls a subroutine to determine which labels can be next.

We claim that if A is ever on the stack (each label of an $\equiv^{\text{loc}}_2$ class mentioned in A is a *legal extension* of the preceding sequence) and the last label is not $'de'$, then then A is the assignment $I^{\text{left}}_{\{2\}}(\lambda)$ for some linear order λ. Conversely, we claim that if A is ever on the stack and its last label is $'de'$, then A is not the assignment $I^{\text{left}}_{\{2\}}(\lambda)$ for some linear order λ, and if m is a label which is not a *legal extension* of A, then appending m to A produces an inconsistent sequence.

First, suppose the last label of A is $'de'$. That label corresponds to a sequence of elements limiting to the left, each of which has some $\equiv^{\text{loc}}_2$ classes $x_0, x_1 \ldots$, so if A were $I_{\{2\}}(\lambda)$ for some λ, A should label the least elements of type $x_0, x_1, \ldots$, so A should end $'de' = {}'dx'_0 = {}'dx'_1 \ldots$. To be precise, suppose $B = I^{\text{right}}_{\{2\}}(\lambda)$ so that $A < B$ is $I_{\{2\}}(\lambda)$. Now A and B should label the same $\equiv^{\text{loc}}_2$ classes. Since the last label in A is $'de'$, A labels no $\equiv^{\text{loc}}_2$ classes, so in the initial state of the local linear consistency game is $(de, \emptyset, B)$. As we argued in the enumeration of $\equiv_3$ classes, this is inconsistent, since $de < B$ and de requires a sequence of elements to descend from above, a terminal segment of which is $< B$, hence in the gap.

The other $A = I^{\text{left}}_{\{1,0\}}$ which can be on the stack when we first read step 6 is $le - lf - le$. This is consistent. Let $B = ge - gf - ge$, $A < B$, $U = \emptyset$; the triple $(le, \emptyset, ge)$ is consistent, since le can refer to the right end and ge can refer to the left end.

The subroutine *Legal Extension* announces in steps 5 and 6 which labels can follow A. For instance, $le - le - le - l(ge, de) - l(ae, de)$ is inconsistent, since the fourth label requires something to limit to it from the right, and nothing is available yet to make up that limiting set. On the other hand, $le - le - le - l(ge, le) - l(ae, de)$ is consistent, since the fourth label requires a successor, which can have $\equiv^{\text{loc}}_2$ type (ge, le) and the fifth label requires something to limit to it from the left, which can be an infinite sequence of elements in $\equiv^{\text{loc}}_2$ class (ge, le). If A labels no $\equiv^{\text{loc}}_2$ classes, and m is a label, then m can follow A just in case the triple $(A, \emptyset, m)$ is consistent. [17] Now suppose A contains already some labels in I^{left}_2. In the next triple (A, U, m), U will contain only $\equiv^{\text{loc}}_2$ types labeled in A. Steps 1 through 4 describe possibilities for U. Step 2: A *predecessor* is an element of U that player II can play, in response to player I's play at g in $(?, U, g)$. Step 3: A *successor* is an element of β that player II can play in response to l in $(l, U, ?)$. Step 4: A *repeatable* subset of U is a set U_0 of $\equiv^{\text{loc}}_2$ classes such that for each $\tau \in U_0$, player II can answer τ in $(?, U, \tau)$ or in $(\tau, U, ?)$ with another element of U_0 and such that (d, U_0, a) is consistent – i.e., U_0 can form infinite sequences

[17] For that reason, $'le'$ can be a final element of A and $'de'$ cannot be a final element of A –$\emptyset$ cannot follow $'de'$.

which ascend to a and descend to d. A repeatable element satisfies every need it creates during the local linear consistency game. Step 4 claims that $\{(ge, le)\}$, $\{(ae, de)\}$, and $\{(ae, le), (ge, de)\}$ have this property and that any repeatable set contains one of those. That these sets are repeatable is clear. Conversely, the only sets which contain none of $\{(ge, le)\}$ or $\{(ae, de)\}$ or $\{(ae, le), (ge, de)\}$ are: $\emptyset$, $\{(ae, le)\}$, and $\{(ge, de)\}$. Clearly, $\emptyset$ doesn't contain anything that can be repeated infinitely and descend to d or ascend to a. $\{(ae, le)\}$ requires an immediate successor, and doesn't contain one; $\{(ge, de)\}$ requires an immediate predecessor and does not contain one. Steps 5 and 6 of the subroutine *legal extension* match situations, in which player I can force a certain $\equiv_2^{\text{left}}$ or $\equiv_2^{\text{right}}$ class, to requirements on U having a successor or a predecessor or repeatable elements.

Suppose A is on the stack in Step 6 and the last label of A is not $'de'$ nor $'le'$. Then a nonempty set $x_0, x_1, \dots$ of $\equiv_2^{\text{loc}}$ types are labeled in A. Until the first $\equiv_2^{\text{loc}}$ type, it has sufficed to check whether two labels will label the same gap, and to check triples $(l, \emptyset, g)$. Only when we introduce the second, third, and final $\equiv_2^{\text{loc}}$ classes do we consider nonempty triples. We create a simple $Th^{\text{right}}(\lambda)$ to pair with A, in order to determine the consistency of A followed by a given next label. We choose: $ax_0 = ax_1 = \dots = ae$ for the right end. If the last element of A is the $\equiv_{\text{loc}}^2$ class (x, y), e.g., $(x, y) = (ae, de)$, then $(y, \{x_0, x_1, \dots\}, ae)$ is the triple between A and $(\lambda, \emptyset)$, the common image of all labels in the right end. A is only consistent if this triple is consistent.

- Step 7 is justified because the initial labels of $I_{\{2\}}^{\text{left}}$ are irrelevant in every further situation where we use $I_{\{2\}}^{\text{left}}$.

- Step 8: We must only pair assignments $I_{\{2\}}^{\text{left}}$ and $I_{\{2\}}^{\text{right}}$ which label the same elements. For, in the definition of the $\equiv_2^{\text{loc}}$-refinement of $I_{\{1,0\}}$, we label every $\equiv_2^{\text{loc}}$ type which is realized in λ because 2 is greater than the indices of $I_{\{1,0\}}$.

- Step 10: Let's check that this process constructs all orderings of the union of the left assignment and the right assignment. For instance, given 12345 and abc, we'd initialize the stack with 12345abc, then find the pair that can be switched, yielding the order 1234a5bc, then find further pairs that can be switched, yielding the orders 123a45bc and 1234ab5c, and again, yielding the orders 12a345bc and 123a4b5c (twice) and 1234abc5, and so on. We claim that every ordering of two sets $A \cup B$ respecting an initial order on A and an initial order on B is found this way: if we want the least element of B to precede n_0-many elements of A, we switch it n_0-many times with the greatest elements of A; then, if we want the next-least element of B to precede $n_1 < n_0$-many elements of A, we switch it n_1-many times with the greatest elements of A; and so on. This routine is inefficient for finding all the orderings of $A \cup B$. But it is efficient for our purposes for two reasons: 1. each switch is illegal just in case it considers switching an element of A and an element of B which label the least and greatest elements of the same $\equiv_n^{\text{loc}}$ class (the least must precede the greatest), and 2. if we consider elements of A and B which label the least and greatest elements of an $\equiv_n^{\text{loc}}$ class, we can enumerate the possibility that those labels are equal, labeling the unique element in that class.

- Step 11: We claim that this pass correctly finds the set of $\equiv_2^{\mathrm{loc}}$ classes α which can be realized in each cut $(b, c) \in I^+_{\{2\}}$, i.e., the set of α such that the least element(s) in α is/are labeled in b and the greatest element(s) in α is/are labeled in c. For each α, we put α in the set *available* when we see the least element(s) in that class, and we take α out of the set *available* when we see the last element(s) in that class.

Now we claim that the subroutine *branching table* correctly describes the $\equiv_4$-different possibilities for the interval $[p, q]$, where $(p, q) \in I^+_{\{2\}}$.

- Step 1: The interval $[p, q]$ could be just the pair $\{p, q\}$, if p is the least or greatest (or unique) element of type (ae, le) or (ge, le) and q is the least or greatest element of type (ge, le) or (ge, de). That happens, as in step 1, just in case p can be a predecessor and q can be a successor. In any other case, the interval must be nonempty – $p = \ 'd\ldots'$ or $p = \ 'lx'$ for $x = (ae, de)$ or $x = (ge, de)$, so that the triple for $[p, q]$ is $(-, U, ?)$.

- For $n < 2$, steps 2 through 4 consider whether an $\equiv_n^{\mathrm{loc}}$-refinement of $I_{\{2\}}$ labels the first element(s) in $\equiv_n^{\mathrm{loc}}$ class α above $p \in I_{\{2\}}$ (and below q). That occurs just in case: $p = \ 'd\ldots'$ or $p = \ 'l\ldots'$ or $p = \ 'g\ldots'$, since the condition $n > 2$ fails, and the final condition – that $p = \ 'a\ldots'$ and there exists a boundary $b < p$ such that α is not realized in (b, p) – fails because $\equiv_n^{\mathrm{loc}}$ is a singleton equivalence class, hence every element of (b, p) realizes α; further, (b, p) is not empty because $p = \ 'a\ldots'$.

- In step 2, if (p, q) is such that both the least elements and the greatest elements there are labeled in a $\equiv_0^{\mathrm{loc}}$ refinement, the interval (p, q) could have < 6 elements so that three refinements exhaust the interval and label some element twice just in case p is $'g\ldots'$ or $'l\ldots'$ or $'o\ldots'$ and labels $\equiv_2^{\mathrm{loc}}$ type (ae, le) or (ge, le), if q is $'g\ldots'$ or $'l\ldots'$ or $'o\ldots'$ (where o means that the least and greatest element are assigned to the same element, a unique element of the describe type) and labels $\equiv_2^{\mathrm{loc}}$ type (ge, de) or (ge, le), and if z is in *Available*. In this case, we add all possible $\equiv_0^{\mathrm{loc}}$ refinements in the cut (p, q) for which some labels are assigned to the same element, $\equiv_1^{\mathrm{loc}}$ refinements of $\equiv_0^{\mathrm{loc}}$ refinements in the cut (p, q) for which some labels are assigned to the same element, and $\equiv_0^{\mathrm{loc}}$ refinements or $\equiv_1^{\mathrm{loc}}$ refinements of $\equiv_0^{\mathrm{loc}}$ refinements in the cut (p, q) for which some labels are assigned to the same element: the least of this is the possibility that $'le > p'$ is assigned to q and $'ge < q'$ is assigned to p or that $'le > p'$ and $'ge < q'$ are assigned to the same element; next-least is: $'oe \in (p, q)'$; largest is possibility that the first two refinements occur without overlap, and then the final refinement assigns the least and greatest element to be equal. The full list is: $p - oe - q, p - le - ge - q, p - le - of - ge - q, p - le - lf - gf - ge - q, p - le - lf - oe - gf - ge - q$. The next-larger assignment, $p - le - lf - le - ge - gf - ge - q$, is a case in which the third element after p has 3 successors, the greatest of which has $\equiv_2^{\mathrm{loc}}$ type (ge, le) and in which the third element before q has 3 successors, the least of which has $\equiv_2^{\mathrm{loc}}$ type (ge, le), and in which the cut $(I^{\mathrm{left}}_{\{1,0\}}[p, \emptyset), I^{\mathrm{right}}_{\{1,0\}}(\emptyset, q])$ happens to contain no $\equiv_3^{\mathrm{loc}}$ classes at all.

- In step 3, if (p, q) is such that the least elements are labeled and the greatest elements are not labeled, then add all $\equiv_0^{\mathrm{loc}}$ refinements on the left of $\equiv_1^{\mathrm{loc}}$ refinements on the left of $\equiv_0^{\mathrm{loc}}$ refinements on the left in (p, q) for which $I^{\mathrm{left}}_{\{1,0\}}[p, \emptyset)$ defines every element between p and q, leaving no $\equiv_3^{\mathrm{loc}}$ elements in the gap $(I^{\mathrm{left}}_{\{1,0\}}[p, \emptyset), q)$, without defining all three refinements and assigning $\equiv_3^{\mathrm{loc}}$ types to them. This is possible just in case p is $'g\ldots'$ or $'l\ldots'$ or $'o\ldots'$ and labels $\equiv_2^{\mathrm{loc}}$ type (ae, le) or (ge, le). Then, if x is in *Available*, add $p - le - \emptyset - q$ to the list of possible $I^{\mathrm{left}}_{\{1,0\}}[p, q]$. The list has one or two elements: if $(ge, de) \in$ *Available*, then $p - le - q$ could exhaust $[p, q]$; if if $(ge, de), (ge, le) \in$ *Available*, then $p - le - le - q$ could exhaust $[p, q]$.

- Step 4 is symmetric.

- We have not considered all possibilities in which p leaves the least element unlabeled or q leaves the greatest element unlabeled– we have only considered the possibilities in which that happens and in which one side can be labeled but its labels don't get refined three times without exhausting the interval (p, q). We must still consider the $\not\equiv_4$ possibilities in which three refinements of the left side of (p, q) and and their $\equiv_3^{\mathrm{loc}}$ class exhaust the gap, while the right side is not labeled, and we also consider the possibility that both left and right ends are unlabeled. We achieve this by making *Stack* the singleton containing the word *unlabeled*, if that side is unlabeled, in Step 5.

- Steps 6,7,8 develop the possible labels on the left end of (p, q) if that end of the interval should be labeled. This completes $I_{\{2,1,0\}}$

- If we find $\equiv_3^{\mathrm{loc}}$ types for the elements discovered in step 8, then we will know $\equiv_3^{\mathrm{loc}}$ types for all elements of $I_{\{2,1,0\}}$.

Let's prove the grouping of $\equiv_3^{\mathrm{loc}}$ classes described in the definition.

Let's introduce a shorthand notation for inextensible $\equiv_3^{\mathrm{left}}$ classes: $'de', 'le - df', 'le - lf - de', 'le - lf - le'$ with the third element in $\equiv_2^{\mathrm{loc}}$ class (ge, de), and $'le - lf - le'$ with the third element in $\equiv_2^{\mathrm{loc}}$ class (ge, le) imply the existence of exactly $0, 1, 2$, or 3 least elements, or 4 or more least elements. So we'll refer to these $\equiv_3^{\mathrm{left}}$ classes as $0, 1, 2, 3, 4$. From this we obtain shorthand notations for $\equiv_3^{\mathrm{loc}}$ class: (n, m) where $n, m < 5$.

Whenever the $\equiv_3^{\mathrm{left}}$ classes $(2, 4)$ is realized, it has 4 or more immediate successors; the first of these is in $\equiv_3^{\mathrm{left}}$ class $(3, 4)$ or $(3, 3)$. In this way, many types imply each other:

1. $(0, 0)$ implies nothing.
2. $(0, 1) \Leftrightarrow (1, 0)$
3. $(0, 2) \Leftrightarrow (1, 1) \Leftrightarrow (2, 0)$
4. $(0, 3) \Leftrightarrow (1, 2) \Leftrightarrow (2, 1) \Leftrightarrow (3, 0)$
5. $(0, 4) \Leftarrow (1, 3) \Leftrightarrow (2, 2) \Leftrightarrow (3, 1) \Rightarrow (4, 0)$
6. $(0, 4) \Leftarrow (1, 4) \Leftarrow (2, 3) \Leftrightarrow (3, 2) \Rightarrow (4, 1) \Rightarrow (4, 0)$

7. $(0,4) \Leftarrow (1,4) \Leftarrow (2,4) \Leftarrow (3,3) \Rightarrow (4,2) \Rightarrow (4,1) \Rightarrow (4,0)$
8. $(3,4) \Rightarrow (4,3) \vee (4,4)$
9. $(4,3) \Rightarrow (3,4) \vee (4,4)$
10. $(4,4)$ implies nothing.

The lines of this enumeration are not an equivalence relation on $\equiv_3^{\mathrm{loc}}$ classes, but they have the following properties: they are a set of sets of $\equiv_3^{\mathrm{loc}}$ classes such that any $\equiv_3^{\mathrm{loc}}$ class is in one of the given sets, the sets are closed under the implication between $\equiv_3^{\mathrm{loc}}$ classes (l,r) and (p,q) which holds in case $\exists x(l^{<x} \wedge r^{>x}) \rightarrow \exists x(p^{<x} \wedge q^{>x})$, any any two sets A and B are distinguished by some $\equiv_3^{\mathrm{loc}}$ class which is in one and not in the other.

If the $\equiv_3^{\mathrm{loc}}$ class (n,m) for $n < m < 3$ exists, its existence is implied by the existence of its neighbor $(n+1, m-1)$. The $\equiv_3^{\mathrm{loc}}$ classes are thereby grouped into between 1 and 7 elements in a short chain of immediate predecessors and successors, and the final three classes.

When the program asserts that N exists, that should be interpreted as saying that one of the groups of $2, 3, 4, 5, 6$ or 7 chains of $\equiv_3^{\mathrm{loc}}$ classes in the list above exists.

- In Step 1, if a $\equiv_3^{\mathrm{loc}}$ type γ exists in the gap $(b,c) \in I_{\{2,1,0\}}$ and implies the existence of an $\equiv_2^{\mathrm{loc}}$ type γ' then (b,c) is a subinterval of $(e,f) \in I_{\{1,0\}}$. So the type γ' must exist in the interval (e,f) and the least element(s) and greatest element(s) of type γ' should be labeled in the $\equiv_2^{\mathrm{loc}}$-refinement of $I_{\{1,0\}}$. Two $\equiv_3^{\mathrm{loc}}$ classes in the same group imply the same sets of $\equiv_2^{\mathrm{loc}}$ classes:
  - $\equiv_3^{\mathrm{loc}}$ classes in 1 imply $\{(ae, de)\}$,
  - $\equiv_3^{\mathrm{loc}}$ classes in 2 imply $\{(ae, le), (ge, de)\}$.
  - $\equiv_3^{\mathrm{loc}}$ classes in N imply $\{(ae, le), (ge, le), (ge, de)\}$.
  - $\equiv_3^{\mathrm{loc}}$ classes in W imply $\{(ae, le), (ge, le)\}$.
  - $\equiv_3^{\mathrm{loc}}$ classes in X imply $\{(ge, le), (ge, de)\}$.
  - $\equiv_3^{\mathrm{loc}}$ classes in Z imply $\{(ge, le)\}$.
- Step 2 is necessary, since if $U = \emptyset$, then in the labeled linear consistency game beginning with no labels ($A = \emptyset$) and $\alpha((\emptyset,\emptyset)) \equiv_3^{\mathrm{left}}$ the $\equiv_3^{\mathrm{left}}$ class that the the main program picked (in step 9.) for the last element of $I_{\{1,0\}}^{\mathrm{left}}[p,\emptyset)$ and the $\alpha((\emptyset,\emptyset)) \equiv_3^{\mathrm{right}}$ the $\equiv_3^{\mathrm{right}}$ class that the the main program picked (in the paragraph after step 9.) for the least element of $I_{\{1,0\}}^{\mathrm{left}}(\emptyset,p]$.
- In the labeled linear consistency game beginning with no labels ($A = \emptyset$) and $\alpha((\emptyset,\emptyset))$ an $\equiv_3$ class which implies the existence of $x_0, x_1, \ldots$, player II has lost unless there is an extension of each of those classes in $\beta((\emptyset,\emptyset)) = U$. So Step 3 is necessary.
- The remaining steps, 6 through 11, describe how W and Z and one $\equiv_3^{\mathrm{left}}$ class require a successor, which require a predecessor, which requires a sequence limiting from below or above, and under what conditions these

requirements are satisfied. The linear consistency game makes that sort of reasoning precise.

This notion of "grouping" can be extended to almost locally closed sets. For instance, each of the $\equiv_2^{\text{loc}}$ classes $\{ad, al, gd, gl\}$ requires either an ascending sequence of elements or a greatest predecessor and requires either a descending sequence of elements or a least successor. We can track when those requirements are satisfied. But we could consider, instead, a set of four $\equiv_2^{\text{loc}}$-almost locally closed sets, each of which is always consistent:

- $\{97, 98, 99\} \subseteq \omega$ is an almost-locally closed set in which every element has $\equiv_2^{\text{loc}}$ class gl.
- $\{97, (\omega, \omega \times 2 \setminus \omega), \omega, \omega + 1, \omega + 2\} \subseteq \omega \times 2 \cup (\omega \times 2)^+$ is an almost-locally closed set realizing the two $\equiv_2^{\text{loc}}$ classes gl and al.
- A symmetric subset of $(\omega \times 2)^*$ realizing the $\equiv_2^{\text{loc}}$ classes gl and gd.
- $\{(0, -97), (\{b \in 2 \times \eta : b < (0, 0)\}, \{b \in 2 \times \eta : b \geq (0, 0)\}), (0, 0), (1, 0), (\{b \in 2 \times \eta : b \leq (1, 0)\}, \{b \in 2 \times \eta : b > (1, 0)\}), (0, 99)\} \subseteq 2 \times \eta \cup (2 \times \eta)^+$ is an almost-locally closed set realizing the two $\equiv_2^{\text{loc}}$ classes gd and al.

If we want to completely replace $\equiv_2^{\text{loc}}$ classes by $\equiv_2^{\text{loc}}$- almost locally closed sets, one further item is sufficient:

- $\{-97, (\{b \in \eta : b < 0\}, \{b \in \eta : b \geq 0\}), 0, (\{b \in \eta : b \leq 0\}, \{b \in \eta : b > 0\}), 99\} \subseteq \eta$ is an almost-locally closed set in which every element has $\equiv_2^{\text{loc}}$ class ad.

Similarly, for $\equiv_3$, we can realize all of the $\equiv_3^{\text{loc}}$ classes in each of the first 7 groups in an almost locally closed set of size $n + 4$, containing $n + 2$ elements of $n \times \eta$ and two cuts, where n is $1, 2, \ldots,$ or 7. The remaining $\equiv$ classes are realized in the following almost-locally-closed sets:

- $\{(0, -97), (\{b \in 2 \times \eta : b < (0, 0)\}, \{b \in 2 \times \eta : b \geq (0, 0)\}), (0, 0), (1, 0), (2, 0), (3, 0), (4, 0), (5, 0), (6, 0), (7, 0), (8, 0), (\{b \in 2 \times \eta : b \leq (8, 0)\}, \{b \in 2 \times \eta : b > (8, 0)\}), (0, 99)\}$ is an almost-locally closed set found in the linear order $8 \times \eta$, realizing the $\equiv_3^{\text{loc}}$ classes $\{(0, 4), (1, 4), (2, 4), (3, 4), (4, 3), (4, 2), (4, 1), (4, 0)\}$.
- $\{95, (\omega, \omega \times 2 \setminus \omega), \omega, \omega + 1, \omega + 2, \omega + 3, \omega + 4, \omega + 5, \omega + 6, \omega + 7\}$ is an almost-locally closed set realizing in $\omega \times 2$ the $\equiv_3^{\text{loc}}$ classes $\{(0, 4), (1, 4), (2, 4), (3, 4), (4, 4)\}$, since $(\omega \times 2, \omega + 4) \equiv_3 (\omega \times 2, 95)$.
- A symmetric subset of $(\omega \times 2)^*$ realizes the $\equiv_3^{\text{loc}}$ classes $\{(4, 0), (4, 1), (4, 2), (4, 3), (4, 4)\}$.
- $\{91, 92, 93, 94, 95, 96, 97\} \subseteq \omega$ is an almost-locally closed set realizing in ω the $\equiv_3^{\text{loc}}$ class $(4, 4)$.

Different sets of these almost locally closed sets realize the same set of $\equiv_3^{\text{loc}}$ classes. For instance, the almost locally closed sets in $(\omega \times 2)^*$ and $\omega \times 2$ given as examples above together realize the same elements as the almost locally closed sets in $8 \times \eta$ and ω. Thus, the power set of this set of almost locally closed sets

counts certain consistent sets of $\equiv_3^{\mathrm{loc}}$ classes multiple times, and this should be discounted in an enumeration of $\equiv_4$ classes.

This concludes the proof that definition 4.2.1 defines an enumeration of the $\equiv_4$ classes of linear orders. □

Implementing the program described in the definition in *Perl*, it ran on this laptop in about two minutes and counted 82988077686330 $\equiv_4$ classes of linear orders.

4.3 Infinitary logic

For any linear order λ, let $EF_\lambda(\mu, \pi)$ be the game in which player I chooses an element $a \in \lambda$, players I and II choose elements $m \in \mu$ and $p \in \pi$ (a normal turn of the EF game) and then the players play $EF_{\{b \in \lambda : b < a\}}((\mu, m), (\pi, p))$. After this process has been repeated, and elements $a_i : i \in I$ of λ have been chosen by player I, and pairs $m_i \in \mu, p_i \in \pi$ have been played by both players, then the players play $EF_{\{b \in \lambda : \forall i \in I (b < a_i)\}}((\mu, m_i)_{i \in I}, (\pi, p_i)_{i \in I})$. We call λ the *clock* in this game. If player II wins $EF_\lambda(\mu, \pi)$, then we say $\mu \equiv_\lambda \pi$.

For the rest of this section, suppose that the clock λ is an ordinal. We want to express $\equiv_\lambda$ as a tree of labels of local classes; now that tree will have a rank for every descending sequence $A \in \lambda$ and have infinitely wide ranks.

Definition 4.3.1: If α is an ordinal and μ is a linear order and I is an assignment of labels into $\mu \cup \mu^+$ then the $\equiv_\alpha^{\mathrm{loc}}$-*refinement of* I is the assignment I' which contains I and in addition, for each cut (b, c) in I and for each inextensible $\equiv_\alpha^{\mathrm{loc}}$-equivalence class τ realized in λ between b and c, a label for the least element or elements of type τ if one of the following conditions hold:

1. $b = \emptyset$ or
2. there is a maximal $b' \in b$ (for all $e \in b, b' \geq e$) such that the label of b' begins "$g \ldots$" or "$l \ldots$" or "$d \ldots$" or b' is "$a\tau' \in (e, f)$" for τ' an $\equiv_\beta^{\mathrm{loc}}$ class such that $\beta < \alpha$ or
3. labels $x\tau' \ldots$ where τ' has quantifier rank $\geq \alpha$ are bounded in b by some element of μ or
4. elements $n \in \mu$ such that (μ, n) is in $\equiv_\alpha^{\mathrm{loc}}$ class τ are bounded in b by some element of μ. [18]

In those cases, I' labels the least element in class τ above b with the label "$l\tau \in (b, c)$", or I' labels the cut below the least decreasing sequence of elements in class τ above b with the label "$d\tau \in (b, c)$". 2. Similar conditions determine whether I' labels the greatest element or elements of type τ below c.

The item which allows τ to be defined above $\sup b$ if b is ultimately of low rank (the 3rd item in the preceding definition) does not pass over limits: suppose β is the least element of A and $\alpha < \beta$ and α is a limit and τ is an $\equiv_\beta^{\mathrm{loc}}$ class and "$a\tau \ldots$" is a label in I_A assigned to the cut (b, c) in λ^+. Suppose that for each $\gamma < \alpha$ no $\equiv_\gamma^{\mathrm{loc}}$ class which is realized in λ is realized in b is not bounded in b. It

[18] That is, there is some bounding element m: $\exists m \in \mu((m < \sup b) \wedge \neg\exists n(n \in \mu^{(m, \sup b)} \wedge (\mu, n) \in \tau))$.

does not follow that the $\equiv_\alpha^{\mathrm{loc}}$ type τ is realized in b. Nor does it follow that, if τ is realized in b, then it is not bounded in b. Thus, the least α for which some $\equiv_\alpha^{\mathrm{loc}}$ type's least occurrence above b is defined in the $\equiv_\alpha^{\mathrm{loc}}$ refinement of I_A can be any $\alpha < \beta$.

Let's see how condition 3 (in the preceding enumeration) works in an example.

- For any $n < \omega$ there is an $m < \omega$ such that $\omega^m \equiv_n \omega^\omega$. In particular, the equivalence holds if $n \leq 2m$.
- $\omega^\omega \equiv_\omega \omega^\omega + \omega^\omega$ (So for any ordinal α, $\omega^\omega + \alpha \equiv_\omega \omega^\omega + \omega^\omega + \alpha$).
- $\omega^\omega \times 2 \equiv_{\omega+1} \omega^\omega \times 3$ (So for any ordinal α, $\omega^\omega \times 2 + \alpha \equiv_{\omega+1} \omega^\omega \times 3 + \alpha$).
- $\omega^\omega \times 4 \equiv_{\omega+2} \omega^\omega \times 5$.

Now let's compute the labelings of those last two linear orders. For $n = 4$ or $n = 5$, the assignment $\cup_{n\in\omega} \cup_{A\subseteq n} I_A(\omega^\omega \times n)$ labels every element of the least copy of ω^ω and labels the gap at the right end, $(\omega^\omega \times n, \emptyset)$. The elements of $\omega^\omega \times n$ which are not labeled by $\cup_{n\in\omega} \cup_{A\subseteq n} I_A(\omega^\omega \times n)$ form an interval. Let's name the interval of unlabeled elements in each model $\lambda_0 \subseteq \omega^\omega \times 4$ and $\mu_0 \subseteq \omega^\omega \times 5$ so that $\omega^\omega \times 4 = \omega^\omega + \lambda_0$ and $\omega^\omega \times 5 = \omega^\omega + \mu_0$. The assignment $I_{\{\omega\}}(\omega^\omega \times n)$ labels the least and greatest element of each $\equiv_\omega$ type in λ_0 and μ_0. This labels every element of the second copy of ω^ω in $\omega^\omega \times n$. The elements of $\omega^\omega \times n$ which are not labeled by $I_{\{\omega\}}(\omega^\omega \times n)$ are an interval, too. Let's call that interval λ_1 in one model and μ_1 in the other model, so that $\omega^\omega \times 4 = \omega^\omega + \lambda_0 = \omega^\omega + \omega^\omega + \lambda_1$ and likewise $\omega^\omega \times 5 = \omega^\omega + \omega^\omega + \mu_1$.

For any finite set $A \subset \omega$, for $n = 4$ or $n = 5$, the assignment $I_{\{\omega\}\cup A}(\omega^\omega \times n)$ labels nothing; if the least ordinal not in A is n, no $\equiv_n^{\mathrm{loc}}$ type can be labeled in the interval $(b, c) = (\omega^\omega + \omega^\omega, \emptyset)$ between the elements of $\omega^\omega \times n$ which have been defined by $I_{\{\omega\}}(\omega^\omega \times n)$, because b is an unbounded sequence of labels of quantifier rank ω and $\omega > n$. For each label in $I_{\{\omega\}}(\omega^\omega \times n)$, the $\equiv_{\omega+1}^{\mathrm{loc}}$ class of that label is the same in both models. All intervals (b, c) are empty, since every element is defined, except for the interval $(b, c) = (\omega^\omega + \omega^\omega, \emptyset)$. There, the $\equiv_{\omega+1}^{\mathrm{loc}}$ classes $(Th_{\omega+1}^{\mathrm{right}}(\omega^\omega + \alpha), Th_{\omega+1}^{\mathrm{left}}(\omega^\omega))$ and $(Th_{\omega+1}^{\mathrm{right}}(\omega^\omega + \alpha), Th_{\omega+1}^{\mathrm{left}}(\omega^\omega + \omega^\omega))$ are realized, for all ordinals $\alpha < \omega^\omega$, and nothing else is realized.

The reader who has patiently followed this description of the labels relevant to $\equiv_{\omega+2}$ will now see the role of condition 3: if condition 3 were not required and $I_{\{\omega\}\cup A}(\omega^\omega \times n)$ were to define every element of the third copy of ω^ω, then the interval of elements not defined in $\omega^\omega \times 4$ would not contain the $\equiv_{\omega+1}^{\mathrm{loc}}$ class $(\omega^\omega \times 2 + \alpha, \omega^\omega \times 2)$, whereas the interval of undefined elements of $\omega^\omega \times 5$ would indeed contain that $\equiv_{\omega+1}^{\mathrm{loc}}$ class, indicating that these linear orders would be $\not\equiv_{\omega+2}$.

The four conditions enumerated in definition 4.3.1 prevent the labeling of many $\equiv_\alpha^{\mathrm{loc}}$ classes in many intervals. In the case of finite k, these conditions make the assignment of labels which is relevant to $\equiv_k$ seem to me like an unusual subset of the set of all first and last elements of all types in all intervals. But when the quantifier rank is infinite, defining τ only above b which are not $a\tau$ and which are ultimately of low rank eliminates many labels from the assignment, often reducing the cardinality of the assignment which labels the first and last element of each $\equiv_\alpha^{\mathrm{loc}}$ class in all intervals.

Definition 4.3.2: For any linear order μ, for any finite sets A and B of ordinals in λ, we define $I_A(\lambda)$ before $I_B(\lambda)$ just in case $\sum_{\alpha\in A} 2^\alpha < \sum_{\alpha\in B} 2^\alpha$.[19] Let $I_\emptyset(\lambda) = \emptyset$.

For each ordinal β, for each finite (possibly empty) set A of ordinals all greater than β, let $I_{A\cup\{\beta\}}(\lambda)$ be the $\equiv^{\mathrm{loc}}_\beta$ refinement of $\cup\{I_{A\cup B}(\lambda) : B$ is a finite subset of $\beta\}$.

The reader should notice that if $\beta \neq 0$, $I_{\{\beta\}}(\lambda)$ is the $\equiv^{\mathrm{loc}}_\beta$ refinement – not of $\emptyset$, but of an already rich set of labels, indeed, the union of labels in the tree up to the rank indexed by $\{\beta\}$.

Lemma 4.3.1: Let $\alpha+1$ be any successor ordinal. Player I has a winning strategy in the game $EF_{\alpha+1}(\mu_0, \mu_1)$ after the first move has identified $a_i \in \mu_i$ if there is a finite subset $A \subseteq \alpha$ such that conditions $1 \wedge 2 \wedge 3$ or $4 \wedge 5 \wedge 6$ hold:

1. for some $\beta < \alpha$, $A \subseteq \beta$, and
2. $I_A(\mu_0)$ and $I_A(\mu_1)$ induce the same ordering $\leq$ on the same labels and the elements a_0 and a_1 are in the same cut $(b, c) \in (I_A(\mu_i))^+$, and
3. some $\equiv^{\mathrm{loc}}_\beta$ type δ (the discrepancy) is realized between $\sup b$ and a_i in μ_i and δ is not realized between $\sup b$ and a_{1-i} in μ, and the least occurrences of δ are definable above $\sup b$ in the sense of definition 4.3.1, or
4. 1 A contains the predecessor, $\alpha - 1$, of α and
5. for each $B \subseteq \alpha - 1$, $I_B(\mu_0)$ and $I_B(\mu_0)$ induce the same ordering $\leq$ on the same labels and there is some cut $(b, c) \in (\cup_{B\subseteq\alpha-1} I_B)^+$ such that a_i and a_{1-i} are between b and c, and the same $\equiv^{\mathrm{loc}}_{\alpha-1}$ classes are realized between b and a_i as are realized between b and a_{1-i}, and
6. for some $\equiv_{\alpha-1}$ class ρ, for some $i < 2$ and $p_i \in \mu_i$ and for all $p_{1-i} \in \mu_{1-i}$ if $I_{A\setminus\{\alpha-1\}}(\mu_0^{>p_0})$ and $I_{A\setminus\{\alpha-1\}}(\mu_1^{>p_1})$ assign the same labels in the same order, then conditions $(1 \wedge 2 \wedge 3) \vee (4 \wedge 5 \wedge 6)$ hold at rank $A \setminus \{\alpha - 1\}$ in the tree of labels after $a_0 \in \mu_0^{>p_0}$ and $a_1 \in \mu_1^{>p_1}$ are played on the first move in the game $EF_\alpha(\mu_0^{>p_0}, \mu_1^{>p_1})$.

Proof: Suppose there exists a finite set $A \subseteq \alpha$ such that conditions $1 \wedge 2 \wedge 3$ hold. Player I plays the element of $\equiv^{\mathrm{loc}}_\beta$ type δ (the discrepancy) between $\sup b$ and a_i in μ_i. Player II must respond with an element of type δ, since β-many moves will remain after this turn is completed (player I could begin the next turn by choosing $\beta < \alpha$ to be the "number of moves" remaining). By assumption, player II will only find such an element below $\leq b$ or above $\geq c$ in μ. But if this was a winning second move in $EF_{\alpha+1}$, then it was a winning first move in $EF_\alpha(\{a \in \mu_i : a < a_i\}, \{a \in \mu_{1-i} : a < a_{1-i}\})$. This contradicts theorem 4.3.3.

Suppose there exists a finite set $A \subseteq \alpha$ such that conditions $4 \wedge 5 \wedge 6$ hold. Player I plays so that on the second move the players have identified $p_0 \in \mu_0$ and $p_1 \in \mu_1$ with the properties described in condition 6. If the antecedent of condition 6 fails, then by theorem 4.3.3, player I has a winning strategy in

[19] This is the usual lexicographical ordering on decreasing sequences of ordinals where $A < B$ holds just in case the greatest element of B is larger than any element of A or the greatest elements of A and B are both α, and $A \setminus \{\alpha\} < B \setminus \{\alpha\}$.

the game in which (a_0, a_1) was the first move in the game $EF_\alpha(\{a \in \mu_i : a > p_i\}, \{a \in \mu_{1-i} : a > p_{1-i}\})$, since that move did not respect the labels which are known to $\equiv_\alpha$. So the antecedent of condition 6 holds, and by its conclusion we find that the lemma is repeated in $EF_{\alpha-1}(\{a \in \mu_0 : a > p_0\}, \{a \in \mu_1 : a > p_1\})$, i.e., we can repeatedly drop quantifier rank using conditions $4 \wedge 5 \wedge 6$ and preserve the discrepancy of condition 3. □

Theorem 4.3.1: If player II has a winning strategy in $EF_{\alpha+1}(\mu_0, \mu_1)$, then for each finite $A \subseteq \alpha$,

- $I_A(\mu_i)$ is the same ordered set of labels assigned into $\mu_i \cup (\mu_i)^+$, and
- if player I plays the first move at the image of a label in one model, either player II plays the image of that label in the other model or player I has a winning strategy in the remainder of the game, and
- if $(b, c) \in (I_A(\mu_i))^+$ and player I plays the first move in μ_i between $\sup b$ and $\inf c$, then player II plays in μ_i between $\sup b$ and $\inf c$, or player I has a winning strategy in the remainder of the game.

Proof by induction on $\sum_{\beta \in A} 2^\beta$.[20]

Suppose that for all $B < A$ that player II must respect I_B or lose. So we can define a cut (b, c) in $\cup\{I_B(\mu_i) : B < A\}$ such that the first move (a_0, a_1) is played between $\sup b$ and $\inf c$ in either model. If I_A labels the least element in $\equiv^{\text{loc}}_\beta$ class τ in (b, c), then by the definition of 2^β, there is a cofinal set of B such that $B < A$ and β is not in any of these B. If some $B < A$ is the immediate predecessor of A, then $A = B \cup \{n\} \setminus n$, for n the least natural number not in B.[21] However, it is very possible that no I_B, for B in that set, have added any labels to b. By definition 4.3.1, that happens just in case $b \cap I_{A\setminus\{\beta\}}$ is nonempty (condition 1) and has no greatest element labeled "$x\tau$" for τ an $\equiv^{\text{loc}}_\delta$ class, $\delta \geq \beta$ (condition 3) or $b \cap I_{A\setminus\{\beta\}}$ is nonempty and has a last element labeled "$a \ldots$" in which elements of every $\equiv^{\text{loc}}_\delta$ class are not bounded (condition 2). In one of those cases, we see that I_A would *not* label the least element in $\equiv^{\text{loc}}_\beta$ class τ in (b, c). If condition 3 were violated, then that same cofinal subset of b of quantifier rank $\geq \beta$ means that condition 3 is violated in the $\equiv^{\text{loc}}_\beta$ refinement of $\cup_{B<A} I_B$. If condition 2 were violated in the $\equiv^{\text{loc}}_\delta$ refinement of $I_{A\setminus\{\beta\}}$ for every $\delta < \beta$, it does not hold that each element of every $\equiv^{\text{loc}}_\beta$ type τ is unbounded below $\sup b$.

We say player I has a *winning condition* if after the players have identified a_0 and a_1 the preceding lemma has held, with conditions $4 \wedge 5 \wedge 6$ so that pairs (p_0, p_1) have been played on each turn, preserving the lemma, as though in the game $EF_{\beta+1}(\mu_0^{p_0}, \mu_1^{p_1})$, for some $\beta < \alpha$, the first move had been to identify $a_0 \in \mu_0$ and $a_1 \in \mu_1$. This lemma identifies a *discrepancy* between μ_0 and μ_1 in its 3rd condition, which player I tries to exploit. When the discrepancy, an $\equiv^{\text{loc}}_\beta$ class τ, exists in μ_i in (b, c), player I plays in μ_{1-i} in answer to $g\tau_0 \in (b', c')$ or in answer to $a\tau_0 \in (b', c')$ or if b has no greatest element. In the latter two cases, when player I is to play in b_0, an initial segment of b, player I needs to play above a possibly infinite number of upper bounds in b. Let β_{b_0} be such that

[20] We use that sum to define the function 2^α for infinite ordinals α: 2^α is the least ordinal number greater than $\sum_{\beta \in A} 2^\beta$ for all $A \subset \alpha$.

[21] $\{B : B < A, B = A \setminus \{\beta\} \cup B', B' \subseteq \beta \text{ is finite }\}$ are the final sets B such that $B < A$.

$\equiv^{\mathrm{loc}}_{\beta_{b_0}}$ classes are defined confinally in b_0. All $\equiv^{\mathrm{loc}}_{\gamma}$ classes for $\gamma > \beta$ are definable above b_0. Further, $\equiv^{\mathrm{loc}}_{\gamma}$ classes ρ for $\gamma \leq \beta$ are definable above b_0 if occurrences of ρ are bounded below b_0. Player I will be unable to exploit the discrepancy just in case the following occurs: b_0 is $I_{A\setminus\{\alpha-1\}} \cap b$. There is a discrepancy, an $\equiv^{\mathrm{loc}}_{\beta}$ class δ between $\sup b$ and a_i, and no element of $\equiv^{\mathrm{loc}}_{\beta}$ class δ between $\sup b$ and a_{1-i} in μ_{1-i}. Player I plays $q_{1-i} \in b_0$ in μ_{1-i}, and player II answers in μ_i, and the set of labels corresponding to $b \setminus b_0$ in $I_{A\setminus\{\alpha-1\}}(\mu^{>q}_{1-i})$ is now bounded by an element of $\equiv^{\mathrm{loc}}_{\beta}$ class δ between $\sup b$ and a_{1-i} in μ_{1-i}. As we increase q in b_0 and it passes over various upper bounds defining elements of $b \setminus b_0$, we find that $I_{A\setminus\{\alpha-1\}}(\mu^{>q}_{1-i})$ assigns various labels to the same element as in $b \setminus b_0$, including – all high-rank elements (as soon as we pass the last high-rank element of b_0) and an increasing number of low quantifier-rank classes. But if $q >$ the upper bound of some $\equiv^{\mathrm{loc}}_{\beta}$ class, then it's least realization above q is the same as it's least realization above b_0. All further labels are the same, too. So if player I intends to play in a series of cuts $b_0 < b_1 < \ldots b$, then it is enough to choose some

To determine which of these player I's move p_i must exceed, player I looks ahead to defining the anomaly δ. Player I finds a finite set of

So, some aspects of the induction, such as the claim that *players can play arbitrarily close to a cut* and the argument for player I playing labels "d" and "l" in π and labels "a" and "g" in μ, go through just as in the case where the clock is finite.

We have supposed that $\cup\{I_B : B < A\}$ assigns the same labels into $\pi \cup \pi^+$ and $\mu \cup \mu^+$ and that β is the least element of A. Now suppose that the (b, c) is a cut, $(b, c) \in (\cup\{I_B : B < A\})^+$ in which the $\equiv^{\mathrm{loc}}_{\beta}$ refinement of $\cup\{I_B : B < A\}$ will differ between π and μ. We follow the definition 4.3.1 to see how this could happen. First, suppose $\equiv^{\mathrm{loc}}_{\beta}$ class τ exists in π between $\sup b$ and $\inf c$, but not between $\sup b$ and $\inf c$ in μ. Then player I plays the first move at the realization of τ. For player II to play a realization of τ, player II must play outside the interval (b, c). Player II therefore plays below some element of b or above some element of c. That label and element exist in I_B for some $B < A$. So player II has not respected I_B, and by the induction hypothesis, loses.

Next, suppose that τ is bounded below $\sup b$ by $m \in \pi$ but τ is not bounded below $\sup b$ in μ. Player I plays the first move between m and $\sup b$ in π. By assumption, the second player respects $\cup\{I_B : B < A\}$. Now a winning condition exists, since the type τ exists between the first move and $\sup b$.

That labels of quantifier rank greater than that of τ are cofinal in $\sup b$ in μ but not in π is a property of the order of labels agreed on by $\cup\{I_B : B < A\}(\pi)$ and $\cup\{I_B : B < A\}(\mu)$.

If there is a least element of type τ in (b, c) in π but not in μ, then player I plays that least element, player II plays an element of type τ in (b, c) in μ by the inductive assumption, and now a discrepancy exists – the elements of type τ above $\sup b$ and below the second player's first move in μ, which are absent in the analogous interval of π.

Now to see that I_A has the same ordering on labels in both models, we follow the argument in Theorem 2.1 in *Ehrenfeucht–Fraïssé Games on Linear Orders*, specifically, the itemized list on the last page of that proof. □

Lemma 2.2 of *Ehrenfeucht–Fraïssé Games on Linear Orders* also holds for infinitary assignments: For any linear order π and for any finite set A of ordinals,

for every label m assigned by $I_A(\pi)$ to $a \in \pi$, for any order types α, β, and γ, the assignment $I_A(\pi + \alpha + \gamma + \beta)$ assigns m to $a \in \pi$. On the other hand, every label m in the domain of $I_A(\pi + \alpha + \gamma + \beta)$ which is fixed under varying α and β is, in fact, assigned to π.

Theorem 4.3.2: For any linear order π and any element $a \in \pi$, for any ordinals $\alpha > \beta$, from

- $\cup\{I_A(\pi) : A \subseteq \alpha\}$,
- $Th_\beta^{\text{right}}(\{b \in \pi : b < a\})$ and $Th_\beta^{\text{left}}(\{b \in \pi : b > a\})$, i.e., $Th_\beta^{\text{loc}}(\pi, a)$,

we can construct $\cup\{I_B(\{b \in \pi : b < a\}) : B \subseteq \beta\}$.

As in the case when k is finite, we must truncate local types, i.e., we can't say that every label in $I_B(\{b \in \pi : b < a\})$ appears in $I_A(\pi)$ or the assignment of labels into $Th_\beta^{\text{left}}(\{b \in \pi : b > a\})$, but that any label in $I_B(\{b \in \pi : b < a\})$ is the truncation of some label appearing in the latter assignments. The proof goes through as for finite k. □

Theorem 4.3.3: If $\alpha + 1$ is a successor ordinal, $\pi \equiv_{\alpha+1} \mu$ holds just in case

- for all finite sets $A \subseteq \alpha$, $I_A(\pi)$ and $I_A(\mu)$ assign the same labels to elements and cuts in π and μ in the same order, and
- for each finite set $A \subseteq \alpha$ and for each label e, if $(e, f) \in I_A(\pi)$ and $(e, m) \in I_B(\mu)$ then $f \in \pi$ just in case $m \in \mu$; if both those conditions hold, then for some $\equiv_\alpha^{\text{loc}}$ class τ, $(\pi, f) \in \tau$ and $(\mu, m) \in \tau$, and
- for each cut $(b, c) \in (I_A(\pi))^+$ and cut $(b', c') \in (I_A(\mu))^+$ such that b and b' contain the same labels, for each $\equiv_\alpha^{\text{loc}}$ class τ, there is some element $n \in \pi^{(b,c)}$ such that $(\pi, n) \in \tau$ just in case there is some element $m \in \mu^{(b',c')}$ such that $(\mu, m) \in \tau$.

If B is an unbounded set of ordinals, $\pi \equiv_{\cup B} \mu$ holds just in case, for all $\beta \in B$, $\pi \equiv_{\cup\beta} \mu$.

Proof: This is a corollary of the preceding two theorems. □

Theorem 4.3.4: For any finite $k \geq 1$,

- $\omega + Z \times (2^k - 2) + \omega^* \not\equiv_{\omega+k}^{\text{left}} \omega + Z \times (2^k - 1) + \omega^*$ and
- $Z \times (2^k - 1) \equiv_{\omega+k} Z \times 2^k$.

Proof: The following establish the base case, $k = 1$:

- For any finite number k, a unique $\equiv_k^{\text{loc}}$ class is realized in Z (or $Z \times \lambda$ for any nonempty λ). Call it $\zeta_k = (\phi_k, \psi_k)$.[22]
- For any finite number k, any $\equiv_k^{\text{loc}}$ class realized in ω, Z, or ω^* is either (ϕ, ψ_k) for some ϕ such that $\phi_k \subseteq \phi$ or (ϕ_k, ψ) for some ψ such that $\psi_k \subseteq \psi$.

[22] Translation is an automorphism of the integers.

- For any finite set $A \subset \omega$ and for any linear order λ, $I^{\text{left}}_A(\omega + Z \times \lambda)$ labels a finite subset of ω, since the least element of type ζ_k above any finite subset of ω occurs within ω.

- Let I be the union of $I^{\text{left}}_A(\omega + Z \times \lambda)$ over all finite sets $A \subset \omega$, for some linear order λ. I is independent of λ. I labels every element of ω, since if the element $a \in \omega$ is the least element not labeled and if every element below a is labeled in $I_A(\omega + \lambda)$ and if k is the least number not in A and if the $\equiv^{\text{loc}}_k$ class of a is (ϕ, ψ_k), then $l(\phi, \psi_k) \in (\{b \in \omega : b < a\}, \emptyset)$ labels a in $I_{A \cup \{k\} \setminus k}(\omega + \lambda)$.

- Only one $\equiv^{\text{loc}}_\omega$ type is realized in Z (or $Z \times \lambda$ for any nonempty λ). Call it ζ_ω.[23]

- $\omega + Z + \omega^* \not\equiv_{\omega+1} \omega + \omega^*$ because ζ_ω is realized in $(I, \emptyset)$ in the former model and not in the latter.[24] This inequivalence is $\not\equiv^{\text{left}}_{\omega+1}$ or left-invariant: for any $\equiv_k$ class γ there is some $\equiv_k$ class α and a label m[25] in the domain of both assignments $I^{\text{left}}_\omega(\omega + Z + \omega^* + \alpha)$ and $I^{\text{left}}_\omega(\omega + \omega^* + \alpha)$ such that $\omega + Z + \omega^* + \alpha + \gamma + \beta \not\equiv_{\omega+1} \omega + \omega^* + \alpha + \gamma + \beta$ because ζ_ω is realized in (I, m) in the former model and not in the latter.

- On the other hand, $Z + Z \equiv_{\omega+1} Z$ because all labels are assigned to the cuts at the left and right ends of the linear order, and ζ_ω occurs in both models.

Now we proceed by induction: Suppose $\omega + Z \times (2^k - 2) + \omega^* \not\equiv^{\text{left}}_{\omega+k} \omega + Z \times (2^k - 1) + \omega^*$ has been proven for some k. Let δ be the $\equiv_{\omega+k}$ class of $\omega + Z \times (2^k - 1) + \omega^*$. Now $(Th^{\text{right}}(\delta), Th^{\text{left}}(\delta))$ is an $\equiv^{\text{loc}}_{\omega+k}$ class. Call it $\zeta_{\omega+k}$. Now $\zeta_{\omega+k}$ is not realized in $\omega + Z \times (2^{k+1} - 2) + \omega^*$ because no element of that model has $(2^k - 1)$-many copies of Z to its left and to its right. On the other hand, $\zeta_{\omega+k}$ is realized in $\omega + Z \times (2^{k+1} - 1) + \omega^*$ because $2^{k+1} - 1 = (2^k - 1) + 1 + (2^k - 1)$. If we let $\alpha = Z$, then the greatest element of ω^* is labeled "$l(ge, de) \in (\emptyset, \emptyset)$" in $\omega + Z \times n + \omega^* + \alpha + \gamma$, where (ge, de) is an $\equiv^{\text{loc}}_2$ equivalence class, so that we have proved $\omega + Z \times (2^{k+1} - 2) + \omega^* \not\equiv^{\text{left}}_{\omega+k+1} \omega + Z \times (2^{k+1} - 1) + \omega^*$.

Let's define a shorthand: Write n for the $\equiv^{\text{left}}_{\omega+k}$ class of $\omega + Z \times n + \omega^*$, if $n < 2^k$ and write (m, n) for the $\equiv^{\text{loc}}_{\omega+k}$ class of pairs (λ, a) such that $\lambda = Z \times \mu$ for some μ and there are m-many copies of Z left of a and n-many copies of Z right of a. Now the $\equiv^{\text{loc}}_{\omega+k}$ classes imply one another in the following way:

- $(m, n) \to (m + 1, n - 1)$ and $(m, n) \to (m - 1, n + 1)$ if $m < 2^k - 1$ and $n < 2^k - 1$,

- $(m, n) \to (m, n - 1)$ if $m = 2^k - 1$ and $n < 2^k - 1$,

- $(m, n) \to (m - 1, n)$ if $n = 2^k - 1$ and $m < 2^k - 1$,

- $(m, n) \to (m + 1, n) \vee (m + 1, n - 1)$ if $n = 2^k - 1$ and $m < 2^k - 1$,

[23] $\equiv^{\text{loc}}_\omega$ classes are *types* in the traditional sense of that word – infinite sets of formulas which we hope to realize at a single element of some model – and they are diverse, in general.

[24] We are using theorem 4.3.3.

[25] If we let $\alpha = 1 + Z$, then $m =$ "$l(ge, de) \in (I, \emptyset)$" labels the least element of α. If we let $\alpha = Z$, then $m =$ "$l(ge, de) \in (I, \emptyset)$" labels the greatest element of ω^*.

- $(m,n) \to (m,n+1) \vee (m-1,n+1)$ if $m = 2^k - 1$ and $n < 2^k - 1$.

If $n = 2^k - 1$, then $Th_{\omega+k+1}(Z \times (2^k - 2))$ assigns the labels

$$d(0,n) < d(1,n) < d(2,n) < \ldots d(2^{k-1}-3,n) < d(2^{k-1}-2,n) <$$

$$a(n,2^{k-1}-2) < a(n,2^{k-1}-3) < \ldots a(n,2) < a(n,1) < a(n,0)$$

and doesn't realize $\zeta_{\omega+k}$ in the central gap $(\{d(2^{k-1}-2,n)\}, \{a(n,2^{k-1}-2)\})$. For $m \geq n$, $Th_{\omega+k+1}(Z \times m)$ assigns the same labels, and does realize $\zeta_{\omega+k}$ in the central gap. By theorem 4.3.3, nothing but assignments of labels and the realization of different $\equiv^{\mathrm{loc}}_{\omega+k}$ classes separate $\equiv_{\omega+k+1}$ classes. So in particular, that $Z \times m$ for $m \geq n$ label the same elements and realize the same $\equiv_{\omega+k+1}$ classes between labels implies that $Z \times n \equiv_{\omega+k} Z \times (n+1)$. $\square$

This proof applies theorem 4.3.3. We now sketch a simpler proof: During the Ehrenfeucht–Fraïssé game of length $\omega + k$, if the first k-many moves identify $a_0 \ldots a_{k-1}$ in λ and $b_0 \ldots b_{k-1}$ in μ, then player I has a winning strategy if for some $i < k-1$, $\{a \in \lambda : a_i < a < a_{i+1}\}$ is finite and $\{b \in \mu : b_i < b < b_{i+1}\}$ is not finite, or visa versa. On the other hand, if the interval between a_i and a_{i+1} is not finite, then its $\equiv_\omega$ theory is $Th_\omega(\omega + \omega^*)$, so player II has a winning strategy just in case for all $i < k-1$, $\{a \in \lambda : a_i < a < a_{i+1}\}$ and $\{b \in \mu : b_i < b < b_{i+1}\}$ are finite and equal, or infinite. It is possible to give a proof which is simpler than applying theorem 4.3.3 because it is easy to say in this case what $\equiv_\omega$ classes of intervals exist in the model and how those classes limit the first k-many moves of either player.

4.4 Quantifier ranks $\equiv_\alpha$ on wellorders

For any linear order μ, let $D(\mu)$ be the set of elements $a \in \mu$ such that a is the limit of an infinite sequence of elements tending towards a from the left. I.e., $D(\mu) = \{a \in \mu : (\exists b \in \mu(b < a)) \wedge (\forall b \in \mu(b < a) \to (\exists c \in \mu(b < c < a)))\}$. For a limit ordinal $\delta < \lambda$, we define the δ-th iterate of D, D^δ, to be $\cap_{\gamma<\delta} D^\gamma(\mu)$. For a successor ordinal $\delta = \gamma + 1$, D^δ is the compound function which computes D of $D^\gamma(\mu)$.

$D^\delta(\lambda)$ is definable by the δ-fold iteration of the preceding definition: The sentences ϕ_δ, defined as

$$\phi_{\gamma+1} = (\exists y \phi_\gamma^{<y}) \wedge (\forall y \exists z (y < z \wedge \phi^{<z})),$$

$$\phi_{\sup B} = \wedge_{\beta \in B} \phi_\beta$$

have quantifier rank $2 \times \delta$ and for any ordinal α, $\alpha \models \phi_\delta$ just in case for all ordinals $\beta > \alpha$, $\alpha \in D^\delta(\beta)$.

For δ any ordinal, consider $Th_{2\times\delta}(\omega^\delta)$ and $Th_{2\times\delta+1}(\omega^\delta)$.

Theorem 4.4.1: Two minimal almost locally closed sets – a minimal almost locally closed set which contains 0 and a minimal almost locally closed set A_δ which contains $\omega^\delta \times 99$ – realize all $\equiv^{\mathrm{loc}}_{2\times\delta}$ and $\equiv^{\mathrm{loc}}_{2\times\delta+1}$ classes which are realized in $\omega^{\delta+1}$. Further, if we let A_γ be the minimal almost locally closed set which contains a typical element of $D^\gamma(\omega^{\delta+1}) \setminus D^{\gamma+1}(\omega^{\delta+1})$ (an element a of D^γ is typical if each of its minimal almost locally closed sets is $\equiv_{2\times\delta}$ to a minimal almost locally closed set of $a + \omega^\gamma$), then for any interval in $\omega^{\delta+1}$ there is an initial segment $(A_\gamma : \gamma < \delta_0)$ of $(A_\gamma : \gamma < \delta)$ realizing exactly the set of $\equiv^{\mathrm{loc}}_{2\times\delta}$ classes realized in the interval.

Proof, by induction on δ. The $\equiv^{\rm loc}_{2\times\delta+2}$ class of $a \in \omega^{\delta+2}$ is determined by $Th_{2\times\delta+2}(\omega^{\delta+1}, a)$. This is determined, in turn, by the tree of labels $I_{\leq 2\times\delta}$ and $\equiv^{\rm loc}_{2\times\delta+2+1}$ classes at labels and between labels. By induction on this theorem, for any $j \leq 2\times\delta$, we can replace each $\equiv^{\rm loc}_j$ class by the almost locally closed class A_γ which realizes it and has least γ, and we can replace any set U of $\equiv^{\rm loc}_j$ classes by initial segments of the sequence $(A_\gamma : \gamma < \delta)$. We can't easily list the $\equiv^{\rm loc}_{2\times\delta+2}$ classes because there is a variety of possibilities. But for any a, copies of A_δ either: 0. don't occur below a, or occur below a and 1. there is a last copy of A_δ below a, or 2. are unbounded below a. In those three cases, 0. the $\equiv^{\rm loc}_{2\times\delta+2}$ class of a is realized in any minimal almost locally closed set which contains 0, 1. the $\equiv^{\rm loc}_{2\times\delta+2}$ class of a follows from its $\equiv^{\rm loc}_{2\times\delta}$ class, from the fact that $\omega^{\delta+2}$ is an ordinal, and from the fact that there is a last copy of A_δ below a, and 2. the $\equiv^{\rm loc}_{2\times\delta+2}$ class of a is realized in $A_{\delta+1}$. The theory of ordinals and the fact of whether A_δ occurs below a and then unboundedly below a or not isn't enough to determine the $\equiv^{\rm loc}_{2\times\delta+2}$ class of a, but it is enough to determine the $\equiv^{\rm loc}_{2\times\delta+2}$- almost locally closed set around a. For in case 1, this set must contain that last copy of A_δ below a. In case 2, this set contains the cut which is the supremum of the copies of A_δ below a. Adding the least ordinal above this cut gives $A_{\delta+1}$. In the end, the result is an almost locally closed set from which either 1. a cannot be eliminated, without giving up almost closure, so that a is realized in $A_{\delta+1}$, or 2. a can be given up, in which case its $\equiv^{\rm loc}_{2\times\delta+2}$ class is realized in $A_{\delta+1}$. Similarly, we induct quantifier rank $2\times\delta+3$ from quantifier rank $2\times\delta+1$. □

The following theorem can be proved directly, by showing that for any clock $\beta < \alpha$, for any element of one model there is an element of the other model such that the same theorem holds in either direction with the clock β. The details seem to be inevitably gory.

By the preceding theorem, on the first move in $EF_{2\times\beta+1}$ or $EF_{2\times\beta+2}$, player I can distinguish locally a variety of $\equiv^{\rm loc}_{2\times\beta}$ classes or $\equiv^{\rm loc}_{2\times\beta+1}$ classes, but the minimal almost locally closed set containing that local class is $\equiv_{2\times\beta+1}$ or $\equiv_{2\times\beta+2}$ to A_γ, for some $\gamma < \beta$. These are the only definable sets of ordinals that player I can use to choose a first move. Player I can 1. show that D^δ is nonempty in one model and not in the other, or 2. show that D^δ has a greatest element in one model and not in the other, or 3. show that the finite part of $D^\delta(\mu_i) = \omega\times\pi_i + n_i$ is different in the two models ($n_0 \neq n_1$).

Theorem 4.4.2: For any ordinals δ and x, let

$$\pi_{i,\delta} = \{y \in \mu_i : y + \omega^\delta > \mu_i\},$$

which is empty or has as its least element $\omega^\delta \times x_\delta$.

For any ordinals α, μ_0, μ_1, $\mu_0 \equiv_\alpha \mu_1$ holds just in case, for each $\delta < \alpha$, the following hold in both μ_i or fail in both μ_i:

1. $\exists\beta(2\times\delta \leq \beta < \alpha)$ and $(\delta > 0 \wedge \omega^\delta < \mu_i) \vee (\delta = 0 \wedge 0 < \mu_i)$,
2. $\exists\beta_0\exists\beta_1(2\times\delta \leq \beta_0 < \beta_1 < \alpha)$ and $\exists x < \mu_i(\mu_i = \omega^{\delta+1}\times x + \pi_{i,\delta})$ and
$$(\pi_{i,\delta} = \emptyset) \vee (\pi_{i,\delta} \setminus \{\omega^{\delta+1}\times x\}) \not\equiv_{\alpha-2} \omega^{\delta+1} + \pi_{i,\delta}, \text{and}$$
3. $\chi_\delta(\mu_0) = \chi_\delta(\mu_1)$ or $\chi_\delta(\mu_0)$ and $\chi_\delta(\mu_1)$ are both $\geq 2^{\alpha-(2\times\delta)} - 1$,

where $\chi_\delta(\mu_i)$ is:

- The number of x such that $\omega^\delta \times x < \mu_i$ and $\omega^\delta \times x + \omega^{d+1} > \mu_i$,
- $+3$ if $\omega^{d+1} < \mu_i$,
- -1 if $\delta > 0$ and $\mu_i < \omega^{\delta+1}$,
- -1 if $\pi_{i,\delta} \neq \emptyset$ and $(\pi_{i,\delta} \setminus \{\omega^\delta \times x_\delta\}) \not\equiv_{2\times\delta} \omega^\delta + \pi_{i,\delta}$.

Further, if one of those final three summands applies to μ_i and not to μ_{1-i} and this difference renders $\xi_\delta(\mu_i)$ small and equal, then there is another $\delta' \neq \delta$ which witnesses $\mu_0 \not\equiv_\alpha \mu_1$.

Proof: The three conditions correspond to properties of μ_i of quantifier-rank α, describing three properties of the elements of $D^\delta(\mu_i)$ which are not individually definable in $\equiv_\alpha$:

1. $D^\delta(\mu_i) \neq \emptyset$. The formula $\phi_\delta^{<x}$ has quantifier rank $2 \times \delta = \beta < \alpha$, so the sentence $\exists x(\phi_\delta^{<x})$ has quantifier rank α. In the game $EF_\alpha(\mu_0, \mu_1)$, player I plays $\omega^\delta \in \mu_i$, then proceeds to verify $\omega^\delta \models \phi_\delta$.
2. If this condition holds, the Cantor normal form of μ_i has no term with exponent δ, and $\pi_{i,\delta}$ is small enough that we can express this fact with a formula ϕ_δ^+ which is similar to ϕ_δ and says $\forall x \in D^\delta(\mu_i)\exists y \in D^\delta(\mu_i) y > x$ and $y \notin \pi_{i,\delta}$.
3. The set of elements of $D^\delta(\mu_i)$ which are not individually definable in $\equiv_\alpha$ are pseudo-finite – i.e., we replace all of $D^\delta(\mu_i)$ before and including the last element of $D^{\delta+1}(\mu_i)$ with 3 elements of $D^\delta(\mu_i)$ and play EF as though $D^\delta(\mu_i)$ were finite sets. Hence the upper bound $2^{\alpha-(2\times\delta)}$ on the definably-sized sets $D^\delta(\mu_i)$.

We should explain the summands of χ_δ.

- If δ is not the largest exponent in the Cantor normal form of μ, the infinite set of elements of $D^\delta(\mu_i)$ less than some element of $D^{\delta+1}(\mu_i)$ are equivalent to 3 elements of $D^\delta(\mu_i)$, since when $2 \times \delta + 2$ moves are left, this infinite subset of $D^\delta(\mu_i)$ can be confused with $le_0 < le_1 < le_2$, the three elements that $I^{\text{left}}_{<2}(D^\delta(\mu_i))$ defines.
- The least multiple of ω^δ with no multiple of $\omega^{\delta+1}$ beyond it is, in fact, the last multiple of $\omega^{\delta+1}$, if that exists. This point is available to be played, unless there are absolutely no multiples of $\omega^{\delta+1}$ in μ_i at all, i.e., δ is maximal. On the other hand, if $\delta = 0$, then the element 0 is playable. Just in case $\delta > 0$, the least multiple $\omega^\delta \times 0$ of ω^δ is not playable as an element of $D^\delta(\mu_i)$.
- If this condition holds, then the last element of $D^\delta(\mu_i)$ is individually definable because it is close to the right end.

If these three conditions fail, then for the same δ $(2 \times \delta < \alpha)$, $D^\delta(\mu_i)$ is nonempty (condition 1); for the same δ $(2 \times \delta < \beta_0 < \alpha)$ $D^\delta(\mu_i)$ has a final element; and the elements of $D^\delta(\mu_i)$ which are not individually definable in $Th_{2\times\delta}(\mu_i)$ as 0 or as the final element of $D^\delta(\mu_i)$ which is definably close to

the right are either equinumerous or numerous enough that $\equiv_{\alpha-(2\times\delta)}$ cannot count them. By the preceding theorem, player I's first move will be locally trivial, in the sense that $\mu_i \in \omega^{\mu_i}$, and interpreting player I's move in ω^{μ_i}, it will be equivalent to something in the minimal almost locally closed set of 0 or of the minimal almost locally closed set containing a typical element of $D^\delta(\omega^{\mu_i}) \setminus D^{\delta+1}(\omega^{\mu_i})$. Player I in fact has a variety of first moves just in case the ordinal μ_i has a complicated Cantor normal form, so that different extensible local types occur close to the right end of μ_i. The deficiencies of those local types are revealed by adding ω^δ to the $\equiv^{\text{loc}}_\beta$ class of the first move of player I, for various δ, and checking whether a smaller $\equiv^{\text{loc}}_\beta$ class results. Thus, the only difference that player I can find between μ_0 and μ_1 is a different number of elements in D^δ after the last limit element of that set (if it is infinite). In summary: the final three summands of χ_δ should be intuitive: 1. the infinite initial part of $D^\delta(\mu)$ will be encountered when there are $\equiv_{2\times\delta+2}$ moves left, at which time it plays the same as three undefinable elements of $D^\delta(\mu)$. 2. If $\delta > 0$ is maximal in the Cantor Normal form of μ, then the first copy of ω^δ contributes to $D^\delta(\mu)$ the element 0, which is $\equiv_1$-definable, and hence useless when counting $D^\delta(\mu)$. 3. A final element of $D^\delta(\mu)$ – the initial element of $\pi_{i,\delta}$ is useless when counting $D^\delta(\mu)$ if the set of elements of $\pi_{i,\delta}$ greater than it is $\equiv_{2\times\delta}$-definable. □

A corollary of the preceding theorem is an enumeration of $\equiv_k$ classes, for finite k: For any ordinal μ, let $D(\mu)\setminus\{$the least element of $D(\mu)$, if $D(\mu)$ contains any element$\} \setminus \{$a definable greatest element, if there is one$\} = D^+(\mu)$.

Theorem 4.4.3: For any ordinal α, the $\equiv_{2+\alpha}$ classes of ordinals can be enumerated by enumerating the $Th_\alpha(D^+(\mu))$ and the following:

- If $Th_\alpha(D^+(\mu)) = 0$ then $\mu \in \{0 \dots 2^{2+\alpha} - 2\}$ or $\mu \equiv_{2+\alpha} 2^{2+\alpha} - 1$ is finite or $\mu \in \{\omega, \omega+1, \omega+2, \omega+3\}$.
- If $\exists\pi(Th_\alpha(D^+(\mu)) = \pi + 1)$ then $\mu = \omega \times \pi + n$ for $n \in 4 \dots 2^{2+\alpha} - 5\}$ or $\mu \equiv_{2+\alpha} \omega \times \pi + 2^{2+\alpha} - 4$ or $\mu = \omega \times \pi + \omega + n$ for $n \in \{0, 1, 2, 3\}$.
- If $Th_\alpha(D^+(\mu))$ is a limit ordinal π, then $\mu = \omega \times \pi + n$ for $n \in \{0, 1, 2, 3\}$.

Furthermore, each $\equiv_\alpha$ class has one extension into an $\equiv_{2+\alpha}$ class which describes a limit ordinal, and these exhaust the $\equiv_{2+\alpha}$ classes which describes a limit ordinal.

Proof: Apply the previous theorem with $\delta = 1$ throughout. When we add $n \in \{0, 1, 2, 3\}$, then either $n = 0$, in which case $\pi_{i,1} = \emptyset$, or $n > 0$, in which case $\pi_{i,1}$ is definable. In these four cases, $\pi_{i,1}$ adds no $\equiv_2$-undefinable final element to $D(\mu)$. When we add $n \in \{4 \dots 2^\alpha - 2\}$, then we do add a final $\equiv_2$-undefinable final element to $D(\mu)$, and this is taken into account in the enumeration. □

So, writing $e(k, WO)$ for the number of $\equiv_k$ classes of wellorders,

$$e(k, WO) = e(k-2, WO) \times (2^k - 3) - e(k-4, WO) \times (2^k - 7) + 7.$$

For instance, there are two $\equiv_1$ classes (let $\alpha = 1$). We enumerate the $\equiv_3$ classes of ordinals as:

- $\{0 \dots 7, \omega, \omega+1, \omega+2, \omega+3\}$ extending $Th_\alpha(D^+(\mu)) = 0$ and

- $\omega + 4$ and $\omega + \omega + n$ for $n \in \{0, 1, 2, 3\}$ extending $Th_\alpha(D^+(\mu)) = 1$.

Similarly, there is a single $\equiv_0$ class of linear orders and there are five $\equiv_2$ classes of ordinals, so the formula indicates $5 \times 13 - 2 = 63$ $\equiv_4$ classes of ordinals. 20 of them have $D^+(\mu) = 0$, 13 of them have $D^+(\mu) = 1$, 13 of them have $D^+(\mu) = 2$, 13 of them have $D^+(\mu) = 3$, and 4 of them have $D^+(\mu) = \omega$. The $\equiv_4$ classes of ordinals are $\{\omega \times \alpha + n : \alpha \in A, n \in N\}$ as (A, N) range over the following set:

- $A = \{0\}, N = \{n\}$, for $n = 0..14$;
- $A = \{0\}, N = \{n : n \geq 15\}$;
- $A = \{1\}, N = \{n\}$, for $n = 0..11$;
- $A = \{1\}, N = \{n : n \geq 12\}$;
- $A = \{2\}, N = \{n\}$, for $n = 0..11$;
- $A = \{2\}, N = \{n : n \geq 12\}$;
- $A = \{3\}, N = \{n\}$, for $n = 0..3$;
- $A = \{\alpha : \alpha \geq 3\}, N = \{n\}$, for $n = 4..11$;
- $A = \{\alpha : \alpha \geq 3\}, N = \{n : n \geq 12\}$;
- $A = \{\alpha : \exists \beta : \alpha = 3 + \beta + 1\}, N = \{n\}$, for $n = 0..3$;
- $A = \{$ limit ordinals $\}, N = \{n\}$, for $n = 0..3$;

The $\equiv_3$ class containing most random Cantor normal form polynomials has as its smallest member $\omega + 4$. So we could say that $\omega + 4$ is $\equiv_3$-typical. A more precise definition is that $\omega + 4$ is the smallest Cantor normal form polynomial such that increasing any coefficient (even adding a term which is not there) leaves it $\equiv_3$. The typical $\equiv_4$ ordinal is $\omega \times 3 + 12$, and the typical $\equiv_5$ polynomial is $\omega^2 + \omega \times 4 + 28$.

Theorem 4.4.4: For $k \geq 6$, $e(k, WO) = 2^{q(k) - \epsilon_k}$, where $q(k) = (k+1) \times (k+1)/4$ if k is even, $q(k) = (k + 2) \times k/4$ if k is odd, and $\epsilon_k \in (0.20, 0.37)$.

Proof: This follows from iterating the formula above – $e(3, WO) = 2 \times 5 - 0 + 7$ (if we set $e(-1) = 0$), and $e(4, WO) = 5 \times 13 - 1 \times 9 + 7 = 63$, and $e(5, WO) = 17 \times 29 - 2 \times 25 + 7 = 450$; $e(6, WO) = 63 \times 61 - 5 \times 57 + 7 = 3565$; $e(7, WO) = 450 \times 125 - 17 \times 121 + 7 = 54200$. For $k = 0, 1, 2, 3, 4, 5, 6, 7$, $\log_2(e(k, WO)) = 0, 1, 2.3, 4.1, 6.0, 8.8, 11.800, 15.726$. Thereafter, $\log_2 e(k, WO) \leq k + \log_2 e(k - 2, W =)$, and $\log_2 e(k, WO) \geq \log_2 e(k - 2, WO) + k - e(k - 4)/e(k - 2, WO) \times \log_e / \log_2$. The sums of all numbers below k and of the same parity is $q(k)$. Finally, $\epsilon < \sum_{k < \infty, k \text{ even}} e(k - 4, WO)/e(k - 2, WO)$ is bounded by a geometric series $\sum (2^{-1/4})^k = 0.841^k$ (so every time k increases by 4, another bit of ϵ is determined), which is bounded by 6.3 times its first term, and $6.3 \times e(4, WO)/e(6, WO) < 0.1604; 6.3 \times e(5, WO)/e(7, WO) < 0.077$ give the bound. □

We compute the limiting ϵ_{even} and ϵ_{odd} by numerically iterating the computation of $e(k, WO)$ and inverting the formula in the preceding theorem:

$$2^{\epsilon_{\text{even}}} = 1.19411673235052; 2^{\epsilon_{\text{odd}}} = 1.23201682615002.$$

Computing $e(k, WO)$ for $k \leq 52$ gives these two values of 2^{ϵ} to 14 digits of accuracy. i.e., we get one more bit in one of the ϵ_i with each increase in k.

4.5 Undecidable linear orders

In [1] we find a linear order λ for which $Th(\lambda)$ is undecidable, though the set of all Σ_n-formulas is computable over λ, for all finite n. These formulas have nested sequences of quantifiers which alternate at most n times between $\exists$ to $\forall$. Counting the number of alternations is different than counting the quantifier rank of a formula – Σ_n corresponds to a game in which player I plays a finite sequence of elements in one model – rather than a single element. The authors of that paper ask whether a simple construction for such a λ exists. Their construction, using iterated dense shuffles, is intuitive. Ordinals are another way to hide some information from Σ_n, and we make a construction on that basis:

Definition 4.5.1: Let $(U_i : i \in \omega)$ be a nested sequence of sets of natural numbers: $U_0 \supseteq U_1 \supseteq U_2 \ldots$. For any $\beta < \omega^\omega$, choose δ maximal such that $\exists x \beta = \omega^\delta \times x$. Now for some ordinal y and some finite number n, $x = \omega \times y + n$. That n is the last Cantor normal form coefficient of β. Let $f(\beta)$ be the nth element of U_δ.

$$\text{Let } \lambda = \sum_{\beta < \omega^\omega} \omega + ((\eta + Z) \times f(\beta)).$$

Now η and Z both have finite axiomatizations in $\equiv_3$, since they realized only one $\equiv^{\text{loc}}_{k-1}$ class, for each $k \geq 3$. The $\equiv^{\text{loc}}_2$ class in η is $\forall\exists$, and the $\equiv^{\text{loc}}_2$ class in Z is $\exists\forall$. We will capture this unfortunate difference in 5, a constant error term:

The following is a $\Sigma_{5+\delta}$ formula: A copy of $((\eta + Z) \times m)$ occurs immediately above the β-th copy of ω for some $\beta \in D^\delta(\omega^\omega)$ just in case $m \in U_\delta$. On the other hand, if we rendered the sequence $(U_i : i < \omega)$ eventually constant, this would not affect $\Sigma_n(\lambda)$ for low n, for Σ_n cannot define $D^{n+1}(\omega^\omega)$, since that set is definable only with $n + 1$-many quantifier alternations. If the sequence U_i is eventually constant and each set U_i is periodic, then $\Sigma_n(\lambda)$ is finitely axiomatizable – For "whenever β is divisible by ω^δ but not $\omega^{\delta+1}$ and there exist n such β' below β and above the last element divisible by $\omega^{\delta+1}$ then there are $f(\beta)$-many copies of $\eta + Z$ before the next copy of ω" is a different sentence of Σ_n for each $f(\beta)$, unless $f(\beta)$ is simply $f(\beta - 1) + k$ for some constant k, in which case we can express the periodic part of U_i with a single formula. If some U_i is not periodic, then any Σ_n which defines $\{0 \in \omega_\beta : \beta \in D^i(\lambda)\}$ is not finitely axiomatizable.

If we choose U_i to be undecidable, then that Σ_n which defines $\{0 \in \omega_\beta : \beta \in D^i(\lambda)\}$ is undecidable. On the other hand, if we choose each U_i to be decidable but the sequence $(U_i : i < \omega)$ to be undecidable (e.g., by diagonalizing that sequence against all programs), then all Σ_n are decidable, but $Th(\lambda)$ is not.

4.6 $\equiv_\lambda$ for λ not wellordered

The sequence of elements $A = (a_i : i \in I)$ of λ which will be chosen during the game EF_λ will decrease in λ as the game progresses; the reversed ordering A^* will be a wellorder. Suppose λ_0 is an initial segment of λ. The same player wins $EF_\lambda(\mu, \pi)$ as wins $EF_{\lambda_0}(\mu, \pi)$, for all linear orders μ and π, if player II wins the *clock-comparison game* between λ and λ_0. Player I plays $a \in \lambda$ and player II responds with $b \in \lambda_0$. After player I has played $a_i : i \in I$ in λ and player II has played $b_i : i \in I$ in λ_0, player I plays an element $a \in \lambda$ such that $\forall i \in I(a < a_i)$, and player II plays an element $b \in \lambda_0$ such that $\forall i \in I(b < b_i)$. The first player who cannot play loses. Now λ_0 may replace λ as a clock if player II have a winning strategy in the clock-comparison game, for then we can translate clock moves in λ into clock moves in λ_0 until λ is exhausted. Informally, player II can survive just as long using λ_0 for a clock as player II can survive using λ for a clock. On initial segments of λ we form equivalence classes: For all $b < a \in \lambda$, we say $b \equiv^{\text{clock}} a$ just in case player II has a winning strategy in the clock-comparison game between $\{c \in \lambda : c < a\}$ and $\{c \in \lambda : c < b\}$.

Lemma 4.6.1: The $\equiv^{\text{clock}}$ classes of λ are wellordered.

Proof: We expand the notion of $\equiv^{\text{clock}}$ to a quasi-ordering on linear orders: $\mu <^{\text{clock}} \pi$ holds just in case the first player wins the clock-comparison game in which the first player plays in π and the second player plays in μ. Now $\mu \equiv^{\text{clock}} \pi$ if neither $\mu <^{\text{clock}} \pi$ nor $\pi <^{\text{clock}} \mu$ – i.e., the second player wins the clock-comparison games of μ versus π and π versus μ. That λ_0 is an initial segment of λ shows $\lambda \not<^{\text{clock}} \mu$.

Suppose $e_i : i \in \omega$ is a descending sequence of $\equiv^{\text{clock}}$ classes of λ. If for some $i \in \omega$, $|e_i| < |e_{i+1}|$, then for any $a \in e_i$ and $b \in e_{i+1}$, player II can win the clock-comparison game between $\{c \in \lambda : c < a\}$ and $\{c \in \lambda : c < b\}$ with the following strategy: Play slowly in e_{i+1}, until player I has exhausted e_i. Then play again in e_{i+1}. Player I will now play in e_j, for $j > i$, a move $\equiv^{\text{clock}}$ or $,^{\text{clock}}$ to the move player II has just played. That equivalence shows how to play the remaining moves in the game. This contradicts the idea that e_i and e_{i+1} are equivalence classes. So for all $i \in \omega$, $|e_{i+1}| \not> |e_i|$. Suppose that there is some $i \in \omega$ such that for all $j \geq i$ $|e_j| = |e_{j+1}|$. Then for any $a \in e_i$ and $b \in e_{i+1}$, player II can win the clock-comparison game between $\{c \in \lambda : c < a\}$ and $\{c \in \lambda : c < b\}$ with the following strategy: play in e_{j+1} while player I is playing in e_j. The players will reach $inf \cup_{i \in \omega} e_i$ together. Then it will be player I's turn to play, and player II can copy and remaining moves. This contradicts the idea that e_i and e_{i+1} are separate equivalence classes. So for infinitely many $i \in \omega$ it holds that $|e_{i-1}| > |e_i|$. Now playing the clock-comparison game in which the smaller clock is an $\equiv^{\text{clock}}$ class is simpler than the general clock-comparison game since on any move after $\{a_i : i \in I\}$ has been played, if player II has not yet lost, then $\{c \in e_i : \forall i \in I c < a_i\} \equiv^{\text{clock}} \{c \in e_i : c < a\}$ for any single element $a \in e_i$. If player I can win this game, the winning strategy cannot depend finely on what element player II plays in e_i, since all of those elements are $\equiv^{\text{clock}}$. That is, player I never finds that $b \in e_{i-1}$ is a winning response to $a \in e_i$, but $b \in e_{i-1}$ would lose as a response to $a' \in e_i$, since a and a' are $\equiv^{\text{clock}}$. A winning strategy which is, in this sense, "blind" to the choices of player II, can only exist if there is an ordinal α such that α^* injects into e_{i-1} and not into e_i. The set U_i of

ordinals which inject into e_i is closed under embedding, i.e, if α embeds into β and β embeds into e_i, then α embeds into e_i. So U_i is in fact the set of all ordinals less than α_i, for $\alpha_i = \sup U_i$. Now if U_i is a decreasing function of i, then $(\alpha_i : i < \omega)$ is a sequence of ordinals which decreases infinitely often. □

Now we turn our attention to the expressive power of play within an $\equiv^{\text{clock}}$ class. For instance, if $a \in \lambda$ and $b \in \lambda$ and $a > b$ and $(\lambda, a) \cong (\lambda, b)$ and f is the isomorphism mapping (λ, a) onto (λ, b), then $b = f(a)$ and $(\lambda, b) \cong (\lambda, f(b))$ and likewise $(\lambda, f^n(a)) \cong (\lambda, f^{n+1}(a))$ for all finite numbers n. So the game $EF_{\{b\in\lambda:b<a\}}$ presents player I with, at least, a string of ω-many initial moves, each of which has the same descriptive power. Indeed, if $a \in \lambda$ and $b \in \lambda$ and $c \in \lambda$ and $a > b > c$ and $\equiv_{\{d\in\lambda:d<a\}}$ is the same as $\equiv_{\{d\in\lambda:d<c\}}$, then the following hold: First, by monotonicity, $\equiv_{\{d\in\lambda:d<b\}}$ is equal to those two equivalence classes. Second, $\equiv_{\{d\in\lambda:d<a\}}$ describes the first and last elements of each equivalence class in $\equiv_{\{d\in\lambda:d<c\}}$, which is $\equiv_{\{d\in\lambda:d<a\}}$ again.

Rather than define $\equiv^{\text{loc}}_{\alpha^*}$ sets, we directly define which $A \subseteq \lambda \cup \lambda^+$ are *locally closed*; we call $Th_{\alpha^*}(\lambda, x)_{x\in A}$ of such a set its $\equiv^{\text{loc}-\text{closed}}_{\alpha^*}$ class. We then call an $\equiv^{\text{loc}}_{\alpha^*}$ class the pair (A, a) where A is an $\equiv^{\text{loc}-\text{closed}}_{\alpha^*}$ class and $a \in A$ is a chosen element. We don't insist on A being minimal, lest this reduce A to the empty set. The *locally closed* sets of λ is the smallest set LC of subsets of $\lambda \cup \lambda^+$ such that each element of λ is in one, and such that for each element $A \in LC$, each $\beta < \alpha$, each set $A_0 = (a_i : i \in \beta) \subseteq A$, each cut $(b, c) \in (A_0)^+$ and each $\equiv^{\text{loc}}_{(\alpha\setminus\beta)^*}$ class τ 1. if $Th^{\text{loc}}_{\alpha^*}((\lambda, a_i)_{i\in\beta})$ implies the existence of an element of type τ in (b, c) then there is an element of type τ in A, 2. if $Th^{\text{loc}}_{\alpha^*}((\lambda, a_i)_{i\in\beta})$ implies there is a least element of type τ in A between b and c then that element is in A, and 3. if $Th^{\text{loc}}_{\alpha^*}((\lambda, a_i)_{i\in\beta})$ implies there are elements of type τ descending towards b without bound then A contains both 2a. the cut describing the limit of those least elements of type τ in (b, c), and 2b. a descending sequence of elements of $\equiv^{\text{loc}}_{(\alpha\setminus\beta)^*}$ class τ in (b, c), indexed by $(\alpha - \beta)$.

The minimal set $A_0 = \emptyset$ requires an $\equiv^{\text{loc}}_{\alpha^*}$-closed set A to contain, for $(\emptyset, \emptyset) \in \emptyset^+$, an element of each type τ which is implied by $\equiv^{\text{loc}}_{(\alpha\setminus\beta)^*}$. But as that equivalence class is trivial, it implies nothing. So $A_0 = \emptyset$ requires nothing of a local closure. Indeed, the empty set is locally closed. Let A_0 be a singleton, containing $a \in \lambda$ of $\equiv^{\text{loc}}_{\alpha^*}$ class τ_0. If $\alpha \leq 1$, then nothing is required of A so that A_0 is locally closed – indeed, A_0 is its own $\equiv^{\text{loc}}_1$-closure. If $\alpha > 1$, then assign $0 \in 1 = \beta$ to $a \in A_0$ and consider any $\equiv_{(\alpha\setminus 1)^*}$ class τ such that $Th^{\text{loc}}_{\alpha^*}((\lambda, a))$ implies the existence of an element of type τ near a. For instance, if $\alpha = 2$ and we analyze the linear order $\eta + \omega + 3 + Z + Z$ and consider $a = 1 \in 3$, then the $\equiv^{\text{loc}}_2$ class of a determines that a has an immediate successor and an immediate predecessor. So a locally closed set containing a must contain all of 3. $Th^{\text{loc}}_2(0 \in 3)$ determines that $0 \in 3$ is a limit of elements from below, so a locally closed set contains $(\eta + \omega, 3 + Z + Z)$ and contains an element of ω. Closing under immediate predecessors and successors brings us to $0 \in \omega$, so an element of η is in the locally closed set. That element of η has limits from above and below.

So the minimal locally closed set containing $1 \in 3$ contains, for ϕ some homomorphism of Z into η, $\phi \cup \{(\eta, \omega + 3 + Z + Z)\} \cup \omega \cup \{(\eta + \omega, 3 + Z + Z)\} \cup 3 \cup \{(\eta + \omega + 3, Z + Z)\} \cup$ the first copy of Z. This might seem unnecessarily large, especially to a reader who, like me, enjoys the notion of almost locally closed sets – sets that are cut off when they become repetitive. A locally closed

set, on the other hand, must continue propagating until it is closed under its own local Skolem functions. We define an almost-locally closed set in the same way, except that we do not add to A the closure of each $A_0 = (a_i : i < \beta) \subseteq A$, but only close A under one example of each $Th_{(\alpha \setminus \beta)^*}(\lambda, a_i)_{i<\beta}$ class.

If λ is a single $\equiv^{\text{clock}}$ class, then $\mu \equiv_\lambda \pi$ holds just in case the same $\equiv^{\text{loc}}_{\alpha^*}$-closed sets exist in both μ and π, for each ordinal α such that α^* injects into λ.

More generally, we construct $I_\lambda(\mu)$, a tree of labeled elements of μ, by induction on the $\equiv^{\text{clock}}$ classes $(e_i : i < \gamma)$ of λ. For each class e_i in turn except the last one, with $a \in e_i$, we create the tree at rank e_i: above each branch B of the tree passing through ranks $(e_j : j < i)$, for each $\equiv_{\{b \in \lambda : b < a\}}$-class of $\equiv^{\text{loc}}_{\{b \in \lambda : b < a\}}$-closed sets in μ with a single starting element $b \in \mu$ (for a much narrower tree, use almost-locally closed classes), we label the least realization of τ above B – either the least copy of τ in μ (ordered by where they realize the element equivalent to $b \in \mu$, or the gap below which no copy of τ in μ realizes an element equivalent to $b \in \mu$ above B, and above which every element exceeds the element equivalent to $b \in \mu$ in a copy of τ in μ.

Theorem 4.6.1: For any linear orders λ, μ, π, $\mu \equiv_\lambda \pi$ holds just in case I_λ injects the same tree of labels into μ and π, and if for all $a \in \lambda$, the same $\equiv^{\text{loc}-\text{closed}}_{\{b \in \lambda : b < a\}}$ classes are realized in μ and π at each label of I_λ and in each gap between labels of I_λ.

Proof: With the notion of $\equiv^{\text{loc}}_\lambda$ as the $\equiv_\lambda$ type of a $\equiv_\lambda$ locally closed set with a chosen element given just before this theorem, we can define for which $\equiv^{\text{loc}}_\lambda$ classes τ the least occurrence(s) of τ can be defined in an interval (b, c) in the set of labels already defined, using the conditions that definition 4.3.1 gives. Likewise, the tree of labels for a nonwellordered λ can be defined as in definition 4.3.2, of the tree of labels for a wellordered clock λ, since the $\equiv^{\text{clock}}$ classes in λ are wellordered. With those definitions, theorem 4.3.1 holds even if the clock λ is not wellordered, since for every finite (descending sequence) $A = (a_i : i < n) \subseteq \lambda$ the labels which are definable in $I_{\{a \in \lambda : a < a_i\}}$ can in fact be played by player I in a descending sequence, leading eventually to the $\equiv^{\text{clock}}$ class of a_{n-1}, at which we wish to define I_A. If there is a discrepancy in the models at this level, player I can play down the sequence of elements of A, and play a sequence of nested intervals, eventually exploiting the discrepancy as in lemma 4.3.1. Theorem 3.2 can be carried out as in the case of finite k – we can reverse the indices of trees and labels, so that they depend on the leftmost label and describe the rightmost, or so that they depend on the rightmost and describe the leftmost. □

If $\lambda < \kappa$ are infinite cardinals, is $\equiv_{\kappa^*}$ a strict refinement of $\equiv_{\lambda^*}$? Of course, $\equiv_{\kappa^*}$ can express the sentence ϕ_κ = "there exists a decreasing sequence of κ-many elements." In [2] we find a construction of large, $\equiv_\lambda$ linear orders, one of which satisfies ϕ_κ and one of which does not, for λ any linear order which does not satisfy ϕ_κ. These can be constructed by iterated application of a rule like that creating the "surreal" numbers, or by the exponentiation of linear orders. Our criterion can guide the construction of linear orders which are $\equiv_\lambda$, even though they are not highly homogeneous, and of linear orders close to the "watershed" between linear orders which are $\equiv_\lambda$ and those which are not.

References

[1] John Chisholm and Michael Moses, *An Undecidable Linear Order That Is n-Decidable for All n*, Notre Dame J. Formal Logic Volume 39, Number 4, 519-526, 1998.

[2] J. Oikkonen, *Undefinability of κ-wellorderings in $L_{\infty\kappa}$* Journal of Symbolic Logic 62:3 999-1020, 1997.

5. MINIMALITY CONSIDERATIONS FOR ORDINAL COMPUTERS MODELING CONSTRUCTIBILITY

with Professor Peter Koepke, Mathematisches Institut, Universität Bonn.

Abstract

We describe a simple model of ordinal computation which can compute truth in the constructible universe. We try to use well-structured programs and direct limits of states at limit times whenever possible. This may make it easier to define a model of ordinal computation within other systems of hypercomputation, especially systems inspired by physical models.

We write a program to compute truth in the constructible universe on an ordinal register machine. We prove that the number of registers in a well-structured universal ordinal register machine is always ≥ 4, greater than the minimum number of registers in a well-structured universal finite-time natural number-storing register machine, but that it can always be kept finite. We conjecture that this number is four. We compare the efficiency of our program which computes the constructible sets to a similar program for an ordinal Turing machine.

5.1 Computation at a limit time – continuity and loops are enough.

Let an ordinal computer be a register machine in the sense of [8], storing ordinals and running for ordinal time. Abstract computation, which puts non-integer register values into the registers of a computer, was pursued in [2] and recently in [9]. Ordinal runtimes were described in [1] and [4] and [5], among others. In [6] we related that model of computation to set theoretic notions including the recursive truth predicate and the theory of sets of ordinals. This paper presents ordinal computation from a machine-focused point of view, and considers the structure of algorithms, direct limits, and complexity.

The active command line cannot be a continuous function of time at a limit time, and some registers will not be continuous at limit times. Legal programs, whose `if`-switches are monotonic or are recently computed from monotonically increasing variables, compute their results without any assumption – beyond continuity – on how register values behave at limit times. Well-structured programs (written as loops of loops as in [3]) compute their results without any assumption on how the command control behaves at a limit time, other than that control does not pass out of a loop until the loop's condition is met. We define ordinal register machines to include illegal and illstructured program,

and in Claim 7 we present their "wellstructured," "legal" version definition and prove, by the end of this section, that these two definitions are equivalent.

Definition 5.1.1: An ordinal register machine contains a finite number of registers, each of which can store an ordinal. A program is a numbered list of commands, each of which has one of the following three forms:

- `Zero(`x`)` : Erases the content of register x, leaving 0.
- x `++` : Increments the value of register x.
- `if` $x = y$ `goto` i `else` j : switches line control.

The value of a register x must be a continuous function of time, over any interval of time in which the command `Zero(`x`)` is not executed. In addition, the state (register values and active command) obeys the following rules at limit times:

1. If the command `Zero(`x`)` is executed at each time $t \in T$, then the value of register x at time $\sup T$ is zero.
2. At a limit time λ, command passes to the minimum of the commands which have been active cofinally often before λ.

The last rule is the same as in [5]. In [4], the active command is stored in binary memory; each bit becomes the lim-sup of its previous values at a limit time. On such a machine, we can code the active command so that it becomes the lim-sup or the lim-inf of the commands active previously. However, for well-structured programs, we can replace all requirements, beyond continuity, on how registers behave at limit times by the requirement that any repeating loop should begin again, at a limit time, by first checking the conditional, and then executing the loop again, and so on.

Definition 5.1.2: A program is well-structured if `...` `goto` `...` switches are only used to model the following two commands:

- `if` $x = y$ `(loop)`.
- `for` x `to` y `(loop)` where the command `Zero(`x`)` is not among the instructions in the loop, or
- `while` $x \leq y$ `(loop)`, where the command `Zero(`x`)` is not in the loop, and where it is provable that x will be incremented at least once during the loop.

The loop `for` x `to` z `(`L`)` is defined in terms of `goto` as `1.` `if` $x > z$ `goto 2;` L`;` x`++; if` $x \leq z$ `goto 1; 2.`

The command `for` x `from` 0 `to` z `(`L`)` is `Zero(`x`); for` x `to` z `(`L`)`.

The command `while` $x \leq y$ `(`L`)` is defined from `goto` as `1.` `if` $x > z$ `goto 2;` L`; if` $x \leq z$ `goto 1; 2...`. That x`++` will be executed at least once during the loop L, and that `Zero(`x`)` is not allowed to appear within L assures that x will grow monotonically, and therefore it will eventually reach or exceed the loop bound.

The loop `while` $x = 0$ `(`L`)`, in which x does not necessarily increment during L, is not considered wellstructured programming. Since x need not increment, the loop could be repeated forever, unlike loops which halt at the

fixed points of the normal functions computed by wellstructured programs. `while`$(x = 0)$ `(`L`)` is defined from `goto` as `1. if` $x > 0$ `goto 2;` L`; if` $x = 0$ `goto 1; 2...` where L does not necessarily increment x.

The well-structured programs form the smallest set of programs containing the basic commands `Zero(`x`)` and x `++` of definition 5.1.1 for all register values of x and y, and closed under concatenation and repeating any sequence of wellstructured programs within a wellstructured loop.

A `for` loop increments its index during each loop, and halts when $x = z$. To make this act like traditional `for` loops, we say that the loop is executed one more time once $x = z$ is reached. The for loop must be programmed on the ordinal register machine so that the conditional is checked first.

On the other hand, a program which increments x and then checks whether $x = y$, and halts if so, and then increments y, and repeats those three steps, will never halt, unless at some limit time, control passes to "`if` $x = y$" or "y`++`" rather than to "x`++`." For instance, if the condition on the loop is $f(x, y, z) < g(x, w)$, then at the end of the loop, we compute $u = f(x, y, z)$ and $v = G(x, w)$, and the minimum instruction in the loop is to compare whether $u < v$. When $u = v$, the loop ends, after executing one more time.

However, checking the conditions which could terminate a loop, immediately on reaching any limit times, leads to:

Lemma 5.1.1: Well-structured programs halt.

Proof: By induction on loops, considered as subprograms for which the lemma is proved. During the execution of the loop `for` x `to` y `(loop)`, the register values are all bounded by $c+$ time, for any c which bounds the initial values of the registers. The absolute number of timesteps used, limited by the length of the program and the register values, is a normal function of the loop index, register x, and so at fixed points of this function, the value of x is time. Therefore, on or before the first such fixed point, the condition $x \geq z$ is met. □

Any register which is not erased for a long time, like the index of a loop, becomes frequently equal to the value of absolute time. On the other hand,

Lemma 5.1.2: If registers $\{x_i : i < \omega\}$ are erased cofinally often before limit time λ, then at some time $\leq \lambda$, all the registers are simultaneously zero.

Proof: Let π_0 and π_1 be functions from ω to ω, such that (π_0, π_1) enumerates $\omega \times \omega$. Let $t_0 < \lambda$. If t_{n-1} has been defined, then let t_n be the next time after t_{n-1} when register $x_{\pi_0(n)}$ is erased. For each i, $\sup\{t_n : n < \omega\} = \sup\{t_n : n < \omega$ and $\pi_0(n) = i\}$, so at that time, register i contains the value zero. □

During the execution of a loop, some registers will be erased and others (at least the indices of the loop) will never be erased, from which we define

Definition 5.1.3: Within a loop, call register x *scratch* if the command `Zero(`x`)` occurs; if not, *monotone* if the command x`++` occurs; and *constant* otherwise.

In programs presented in this paper, we will use the symbols `MON` or `SCR` to define a variable to be monotone or scratch in this sense.

We want to make the following program illegal:

- `for` i `from` 0 `to` ω `(Zero(`x`); `x`++);`

- `if` $x = 0$ `(Zero(`y`))`

because it tests the limit of a noncontinuous register value. We want to call x a scratch variable and prevent a variable which, like x, has been erased infinitely often from appearing in the conditional of an `if`-switch until after it has been erased again. Suppose f and g are normal functions of two variables, that can be computed without using the commands `Zero(`x`)` or `Zero(`y`)`. The following program should be legal:

- `for` i `from` 0 `to` ω `(`$x =$ `f(`x, y`); `$y =$ `g(`x, y`));`
- `Zero(`v`);`
- `for` i `from` 0 `to` y `(for` u `from` 0 `to` v `(`v`++));`
- `if` $v = f(x, x)$ `(Zero(`z`))`

We might abbreviate the last two lines as `if` $2^y = f(x, x)$ `(Zero(`z`);)`. In case v were larger than 2^y, we had to erase v before we could increase it monotonically to $v = 2^y$. This should be legal because y is monotone and v depends only on y.

Definition 5.1.4: If a program contains the following:

- a loop A which contains the command `Zero(`x`)`; let X be the first command in the loop A (writing the program using `increment`, `goto`, and `zero` commands, X appears as the earliest command),
- a path B from the command X to the command Y,
- Y performs a switch on the variable x, and
- in the path B the command `Zero(`x`)` doesn't occur,

then the program is illegal.[1]

An illegal program can test the limit of a discontinuous variable. This is because the program could loop infinitely often through the loop A, each time possibly zeroing x, and then switch on the value of x. The switch would then notice whether x is zero after being zero'd (and perhaps increased) infinitely often. We want to write programs independent of the limiting behavior of scratch variables, i.e., independent of condition 1 in definition 1. Using this notion, we can say formally that a scratch variable is not legal in an `if` conditional immediately after it has been incremented and erased infinitely often, but that immediately after it has been erased one more time, it becomes *legal* for use in an `if` conditional. Now we have replaced both conditions 1 and 2 of definition 1 with restrictions on how programs are written, so it turns out that those conditions are not necessary to the proper working of an infinitary machine.

Lemma 5.1.1: The class of computable functions remains the same if, in definition 5.1.1, we require the program to have the form `while(`$x = 0$`) loop`, where `loop` is a wellstructured program obeying the following two programming techniques:

[1] We thank the anonymous referee for suggesting this definition, which simplifes and corrects an earlier definition of illegal programming.

- Explicitly empty all scratch registers at the beginning of the loop.
- Registers used in a switch or the conditional of a loop should depend in a wellfounded way on monotone registers.

The assumptions we need on how a state behaves at a limit time can be relaxed from conditions 1 and 2 in definition 1 to:

1. at a limit time, a wellstructured program evaluates the (unique) active loop, and determines whether to continue looping, and
2. monotone register values pass continuously to their limits (i.e., don't jump) at limit times.

Wellstructured programs always halt, since they compute normal functions and halt when the loop bound is reached by the loop index. Nonwellfounded programs can certainly perform unbounded search. Hence, the `while` loop in the statement of this claim is necessary.

During the rest of this section, we will prove the claim as a generalization of the theorem in [3] reducing all branching programs to loops. That theorem applies only to finite-time computers storing ordinals. However, the theorem still applies if we make the signature (the set of functions and predicates that can be performed instantaneously in a particular model of computation, as in [9] page 322.) include ordinal arithmetic and Gödel pairing.

A finite-time ordinal-storing register machine with only the successor in its signature (an ORM operating for finite time has the successor and Zero in its signature) cannot do arithmetic on its memory elements. The functions of addition, of finding the predecessor of a successor, etc., all take infinite time. However, it is clear that an ordinal register machine can perform these operations of arithmetic, since ordinal addition is iterated succession, multiplication is iterated addition, and exponentiation is iterated multiplication (see details before Lemma 12). Further, the Gödel pairing function, sending (α, β) to the order type of $(\alpha \times \beta, <^g)$, where $<^g$ is the ordering that first compares maxima, then compares lexicographical order (more details in [6], section 2) is also clearly computable by an ordinal register machine that enumerates $\alpha \times \beta$ in the desired order and increments $G(\alpha, \beta)$ with each step.

Definition 5.1.5: Let G be the pairing function taking (a, b) to the order type of pairs $(c, d) <^g (a, b)$, where $(c, d) <^g (a, b)$ iff $\max(c, d) < \max(a, b)$, or $\max(c, d) = \max(a, b)$ and $c < a$, or $\max(c, d) = \max(a, b)$ and $c = a$ and $d < b$.

It is clear how to program the preceding definitions, so ordinal register machines surely compute the signature $\Sigma = (Ord, 0, 1, +, \times, \exp, G, G^{-1})$. Ordinal computers performing finite sequences of operations in that signature form the set of **While**-computable functions over Σ, defined in [9] page 323. That is equivalent to any other reasonable notion of what a finite-time computer could compute, with ordinals in storage and oracles for the functions in Σ.

Definition 5.1.6: Call a finite-time register machine storing ordinals and able to compute the functions in signature Σ in one step an abstract ordinal computer.

Since the signature Σ can code sequences of ordinals as a single ordinal, and since ordinals contain the natural numbers, many natural notions of abstract computability agree over the ordinals, including the interesting machine models described in section 8 of [9].

Proof (of claim 5.1.1): Suppose a model of ordinal computation is proposed, so that on input x it produces output y iff $\phi(x, y, \alpha_0...\alpha_n)$ holds, where ϕ is a Δ_1 concept of set theory. For instance, ϕ might say that there is a computation that starts with x (and the parameters), proceeds according to Δ_0 rules (where $\psi(x, y)$ is Δ_0 if all quantification is bounded to x and y) and ends with a designated output register holding the value y. Then ϕ has a Σ_1 representation. If *any* computation which starts with x and proceeds legally must end with y, then the model has a Δ_1 description. For instance, any reasonable variation on our definition of ordinal computer has a Δ_1 description. We mean to show that all of these can be modeled on our computer. To determine the truth of ϕ, we search through L to find either an example that proves ϕ in its Σ_1 form, or a counterexample that disproves ϕ in its Π_1 form. The `while` loop in the claim 5.1.1 can perform this search, if the following lemma holds.□

Lemma 5.1.3: Wellstructured programs can determine the truth of any Δ_0 sentence, with constant symbols referring to ordinals, of ZFC.

Proof: Fix an enumeration of formulas with ordinal parameters to prove this lemma by induction. The abstract ordinal computers in definition 5.1.6 are Church-Turing complete, so they can compute syntactic operations on formulas, in their codes as ordinals (we will mention this again in definition 16). For instance, we can make the description of ϕ be shallowly accessible in the ordinal coding of ϕ and its parameters $\alpha_1...\alpha_n$ as $G(n_\phi, G(\alpha_0, G(...)))$, where n_ϕ is the Gödel number of the formula. We can choose that the operations $\neg, \wedge, \vee$ increase the Gödel number of a formula, so that to prove the lemma by induction, we only have to deal with formulas $\exists z < x\psi$ or $\forall z < x\psi$. The program corresponding to ϕ has an outer loop `for` z `from` 0 `to` x `loop`, where the loop pushes z into the stack of variables, and then runs the program corresponding to ψ to determines whether ψ holds for that particular value of z. □

5.2 A universal ordinal register machine program

In this section we write a universal program. This improves on Lemma 9 which found a wellstructured program to decide each bounded formula ϕ. The universal L-program reads a code for ϕ and its parameters as input, and determines the truth of ϕ in time at most ordinal-exponential in the size of those parameters. To be precise about the size of the parameters, the reader may wish to check that $G(n_\phi, G(\alpha_0, G(\alpha_1, ...)))$ is $\leq$ the first ordinal of the form ω^{ω^α} which is larger than all of the α_i.

Lemma 5.2.1: $G(\gamma, \gamma) = \gamma$ iff γ is a $\times$-closed ordinal.

Proof: Exercise. Hint (only if) Prove $G(\alpha \times \beta, \alpha \times \beta) > \alpha \times \beta$ unless α or β is 1, by finding a large ordinal product contained in the order type of $<^g$ in definition 5.1.5. Hint (if) Prove by induction that if α-many Cantor-Bendixon derivatives (which "derivative" eliminates all the successor elements) reduce γ

to a finite set, then $\alpha \times 2 + 1$-many Cantor-Bendixon derivatives reduce $G(\gamma, \gamma)$ to the empty set. As a result, every element of $|G(\omega^{\omega^\alpha}, \omega^{\omega^\alpha}), <^g|$ is eliminated by $< \omega^\alpha$-many derivatives. $\square$

Proof (only if): We prove $G(\alpha \times \beta, \alpha \times \beta) \geq \alpha^2 \times (-1 + \beta)$ (by $-1 + \beta$ we mean the ordinal which is β if β is infinite, and $\beta - 1$ if β is finite... it is obtained during our proof as a set isomorphic to β, but missing its first element, hence we write it in this way). Label the elements of $\alpha \times \beta$ as $\{(a, b) : a \in \alpha; b \in \beta\}$. The ordering on the ordinal $\alpha \times \beta$ is $<_l$, the reverse lexicographical ordering: $(a, b) <_l (a', b')$ if $b < b'$ or $b = b'$ and $a < a'$. Now for each $b \in \beta$, b not maximal in β, $G(\alpha \times \beta, \alpha \times \beta)$ gives $S_b := \{((a, b), (a', b+1) : a, a' \in \alpha\}$ its lexicographical order because the pair $((a, b), (a', b + 1))$ achieves its maximum on its second element. G orders S_b as $\alpha \times \alpha$, there are at least $-1 + \beta$ many sets S_b, and G orders the sets S_b in the same order as β orders the pairs $(b, b+1)$. Proof (if): The $\times$-closed ordinals are the ordinals ω^α for various α. We proceed by induction on α. If α is a successor, then $G(\omega^\alpha, \omega^\alpha) = \sum_{n<\omega} G(\omega^{\alpha-1} \times n, \omega^{\alpha-1} \times n)$. Taking Cantor-Bendixon derivatives (passing from a set to the set of its limit points) of that order type α many times leaves the emptyset, so the order type is $\leq \omega^\alpha$. If α is a limit ordinal, $G(\omega^\alpha, \omega^\alpha) \leq \sum_{\beta<\alpha} G(\omega^\beta, \omega^\beta)$ since that sum simply repeats some intervals in the construction of $G(\omega^\alpha, \omega^\alpha)$. But if $c \in \gamma$ is a successor and $b \in \beta$ is a successor, then (c, b) is a successor in $G(\gamma, \beta)$, and, more generally, if c is not in the ϵ-th Cantor-Bendixon derivative of γ, and b is not in the δ-th Cantor-Bendixon derivative of β, then (c, b) is not in the $\max(\epsilon, \delta)$-th derivative of $G(\gamma, \beta)$. Hence, $G(\alpha, \alpha) \leq \omega^\alpha$. $\square$

We will define `Push` and `Pop` on an ordinal register called `Stack` which stores the decreasing sequence of ordinals $\beta > \beta_1 \ldots \beta_{n-1} \geq \beta_n$, where the last two values are allowed to be equal only if β_n is a limit. The elements of this sequence code formulas. The formula coded by β_1 is being considered, to determine whether it witnesses the truth of β. Each β_{i+1} was found while searching for a witness to the truth of β_i, so the sequence is decreasing.

Definition 5.2.1: We code a finite, monotonically decreasing sequence of ordinals $\beta > \beta_1 \ldots \beta_{n-1} \geq \beta_n$, where $\beta_{n-1} \geq \beta_n$ occurs only if β_n is a limit, as $\texttt{Stack} = 2^{\beta+1} + \sum_{i=1\ldots n-1} 2^{\beta_i+1} + 2^{\beta_n}$ if β_n is a limit, and as $\texttt{Stack} = 2^{\beta+1} + \sum_{i=1\ldots n-1} 2^{\beta_i+1} + 2^{\beta_n+1}$ if β_n is not a limit.

We include β_i on the stack as 2^{β_i+1} so that the stack has as its least term the value 2^λ, for λ is a limit ordinal, if and only if that term has been reached as the limit of considering all finite sequences of ordinals $< \lambda$. That is, whenever we `Push` an element β_i onto the stack, the intended exponent is a successor. A final exponent which is a limit only occurs "magically" at a limit time, and indicates that our infinitely-long attempt to prove the formula coded λ is true has failed. So, we conclude λ is false, and go on.

Recall that ordinal multiplication and exponentiation are defined to be continuous in their second term: $\alpha \times (\beta + 1) = \alpha \times \beta + \alpha$, and for X a set of ordinals, $\alpha \times \sup X = \sup\{\alpha \times x : x \in X\}$. (Lemma 14 about `Push` uses this) $2^{\beta+1} = 2^\beta \times 2$ and for X a set of ordinals, $2^{\sup X} = \sup\{2^x : x \in X\}$. On the other hand, 2^β is isomorphic to the set of finite descending sequences of ordinals less that β, ordered lexicographically:

Lemma 5.2.2: $(\beta_i : i < n) \mapsto \sum_{i<n} 2^{\beta_i}$ is an isomorphism between the set of finite, descending sequences of ordinals all less than β, and 2^β.

Proof: We construct the inverse: Given an ordinal $\alpha < \beta$, let β_0 be the supremum of those γ such that $2^\gamma \leq \alpha$. Ordinal exponentiation is continuous, so $2^{\beta_0} \leq \alpha < \beta$. If $\alpha \neq 2^{\beta_0}$, let α_1 be such that $\alpha = 2^{\beta_0} + \alpha_1$. Let $\beta_1 = \sup\{\gamma : 2^\gamma \leq \alpha_1\}$. Again, $\alpha_1 \geq 2^{\beta_1}$. If $\beta_1 \geq \beta_0$, then $\alpha \geq 2^{\beta_0} + 2^{\beta_1} \geq 2^{\beta_0+1}$, contradicting the definition of β_0. So $\beta_1 < \beta$. So continue, to find $\beta_0 > \beta_1 > ... > \beta_n$, such that $\alpha_n = 2^{\beta_n}$. The sequence is finite since β is a wellorder. So the sequence of exponents of α is a finite sequence of ordinals, all $< \beta$. Since $\{\gamma : 2^\gamma \leq \sum_{i<n} 2^{\beta_i}\} = \{\gamma : \gamma \leq \beta_0\}$, we have inverted the summation of a decreasing sequence of ordinals. □

Definition 5.2.2: The program `Push(Stack,` β`)` is the following routine:

```
MON Stack;
SCR i, γ = 0, δ;
for δ from 0 to Stack (
   for (i from 0 to 2^(β+1)) (γ++);
   if (γ > Stack) (for Stack to γ; halt)
)
```

Lemma 5.2.3: `Push(Stack,` β_i`)` increases the Stack to the next full multiple of 2^{β_i+1}.

Proof: So `Push` sets `Stack` equal to $2^{\beta+1} \times \delta$, for δ the least ordinal for which $2^{\beta+1} \times \delta >$ `Stack`. □

Recall how we read the register `Stack` from definition 5.2.1. If β was on the stack (`Stack` $= \sigma + 2^{\beta+1} + \tau$), then `Push` increases τ to $2^{\beta+1}$, leaving `Stack` $= \sigma + 2^{\beta+2}$). I.e., the least element on the stack is changed from β to $\beta+1$. If β was not on the stack (`Stack` $= \sigma + \tau, \sigma = 2^{\beta_i+2} \times \delta$, for some δ, and $\tau < 2^{\beta+1}$), then `Push` increases τ to 2^{β_i+1}, leaving (`Stack` $= \sigma + 2^{\beta+1}$). Pushing β onto a stack erases all stack values less than β.

From `Stack` $= \sum 2^{\alpha_i}$ we will want to read the least exponent α_i which is $\geq$ a certain threshold. We set a "small stack" to be $\tau = \sum_{i>j} 2^{\alpha_i}$, represent the stack as $\sigma + 2^\epsilon + \tau$, and check whether $\epsilon = \alpha_i$ is $>$ than the threshold. Unless α is a limit, we will interpret α as a stack element. If α is a limit, we will only be interested in it, in case it is the predecessor of the next-larger exponent, α', in which case α witnesses that α' is false. We can find the representation `Stack` $= \sigma + 2^\epsilon + \tau$ in many ways, but simply trying all possibilities is one option:

We define two functions `Pop`, one to take the smallest value off the stack, and one to take successive values off the stack:

Definition 5.2.3: `PopLeast(Stack,` β`)` is the following routine:

```
CONSTANT Stack, β;
SCR σ, ε;
for ε from 0 to β + 1 (
   for σ from 0 to Stack (
      if (σ + 2^ε = Stack) ( return ε)
   )
)
```

Definition 5.2.4: `PopNext(Stack, Threshold,` β`)` is the following routine:

```
CONSTANT Stack, Threshold, β;
SCR SmallStack = 0, TempStack = 0, σ, ε, κ;
```

```
for ϵ to β (
   for σ from 0 to Stack (
      if (σ + 2^(ϵ+1)+SmallStack = Stack) (
         if (ϵ ≥ Threshold) (return ϵ);
         for TempStack to Smallstack ();
         for Smallstack to 2^(ϵ+1) ();
         for κ from 0 to TempStack (Smallstack++)
      )
   )
);
return ϵ
```

We intend these programs to be applied when the value of `Stack` is between $2^{\beta+1}$ and $2^{\beta+2}$. In that situation, there is always something on the stack smaller than β. If there were not, then these programs would return nothing, which is reasonable when `PopNext` is designed to find a stack element less than β and larger than a given threshold.

Neither program `Pop` really changes the stack. They just read the least element $\beta_j + 1$ of the stack which is not larger than β, or the least element which is >`Threshold`. We read the whole exponent, $\beta_j + 1$, not just β_j, since the stack might contain, as its last term, 2^λ for λ a limit.

Abstract ordinal computers as in definition 5.1.6 can compute syntactic operations on the codes of formulas, in the signature Σ. We would like to show that ordinal computers can compute, in addition to Σ, the truth predicate T, determining whether any Δ_0 formula is true. Let's gather all of the syntactic formula manipulation into an abstract ordinal program called W for Witnessing, as is done in [6], section 6. We have used this notion already in lemma 9.

Definition 5.2.5: Let $W(\beta, \gamma, T(\gamma))$ be an abstract ordinal computer program that determines whether γ and its truth value $T(\gamma)$ are sufficient information to witness the truth of β. Let the output of W be 0 unless some pair $(\gamma, T(\gamma))$ with $\gamma < \beta$ witnesses the truth of β, in which case W outputs 1. In particular, $W(\beta, \gamma, T(\gamma))$ is the program which finds the syntactic structure of β, and then

- if β codes an atomic sentence with constant symbols for ordinals and for $T(\gamma)$, in the signature $\{<, G\}$, the program W evaluates that atomic sentence.
- if β and γ code the sentences ϕ and $\neg\phi$, $W = 1 - T(\gamma)$.
- if β codes the sentences $\phi \vee \psi$ and γ codes ψ, then $W(\beta, \gamma, T(\gamma)) = T(\gamma)$.
- if β codes the sentences $\exists x < c\phi$, where c is a constant symbol, and if γ codes the sentence $c' < c \wedge \phi(c'/x)$ where the constant c' replaces the variable x, then $W(\beta, \gamma, T(\gamma)) = T(\gamma)$.

Then β is true iff there is some witness $\gamma < \beta$ such that $W(\beta, \gamma, T(\gamma)) = 1$. We will find the truth value of β by searching through decreasing sequences of ordinals $< \beta$, until we find a witnessing sequence, a stack which conveys its witnessing – through pairs of ordinals of the form $\alpha' > \alpha$ such that $W(\alpha', \alpha, T(\alpha))$ holds, or of the form $\alpha' = \alpha + 1$ such that $T(\alpha)$ is known – from a limit ordinal α which appears twice in the stack. This situation arises as the limit of a search

over all ordinal sequences $< \alpha$ during which we did not find a witnessing sequence for α. This is the falsehood from which we conclude, via the witnessing sequence, the truth value of β.

Definition 5.2.6: Say $\beta = \beta_0 > \beta_1 > \beta_2 ... > \beta_{n-1} = \beta_n$ is a witnessing sequence for β if for each $i = 1 \ldots n-1$, $W(\beta_{i-1}, \beta_i, T(\beta_i)) = 1 = T(\beta_{i-1})$ or $\beta_{i-1} = \beta_i + 1$ and $T(\beta_{i-1}) = 0$ and β_n is a limit ordinal and $T(\beta_n) = 0$.

The terminal value of the stack will be $2^{\beta+1} + \sum_{i=1 \ldots n-1} 2^{\beta_i+1} + 2^{\beta_n}$, where $\beta_n = \beta_{n-1}$ is a limit, and no other β_i is a limit. That last summand witnesses that we have examined every possible witness for β_{n-1} and found none, hence β_{n-1} is false. Each summand then witnesses the truth value of the preceding summand, back to β, and we are done. We cannot simply loop through *all* decreasing sequences. If we know that β_j is true, but that β_j doesn't witness β_{j-1}, we must skip the sequence $\cdots \beta_j, \beta_j - 1 \cdots$, since that sequence, as soon as we know the truth value $T(\beta_j - 1)$ and check that $W(\beta_j, \beta_j - 1, T(\beta_j - 1)) = 0$, we intend to interpret to mean that β_j is false. We should only reach that sequence if no $\beta_{j+1} < \beta_j$ could witness that β_j is true. This "skip" is performed by `Push`-ing the Stack to $\sum_{i<j} 2^{\beta_i+1} + 2^{\beta_j+1} + 2^{\beta_j+1}$. Of course, this also speeds up the program: once we know that β_j is true but that that our current witnessing sequence $2^{\beta+1} + \sum_{i=1 \ldots j-1} 2^{\beta_i+1} + 2^{\beta_j+1}$ doesn't witness β's truth, we move on, and consider $2^{\beta+1} + \sum_{i=1 \ldots j-1} 2^{\beta_i+1} + 2^{\beta_j+2}$.

Definition 5.2.7: `Truth(`β`)` is the following program:

```
CONSTANT: β;
MONOTONE: Stack, i;
SCRATCH: α, α′;
Push(Stack,β);
for i from 0 to 2^β (
   α = PopLeast(Stack, β);
   if α is a successor (Stack ++; α = 0);
   α′ = PopNext(Stack, α, β);
   if α′ ≠ α (Push(Stack, α));
   if α′ = α (
   ν = 0; % This is the truth value of α′.
      while α′ ≤ β (
         if α′ = β (return ν);
         α = α′;
         α′ = PopNext(Stack,α + 1,β);
         if W(α′,α,ν)= 0 and α′ ≠ α + 1 (
            α′ = β; % to terminate the while loop
            Push(Stack,α)
         );
         if W(α′,α,ν)= 1 (ν = 1);
         if W(α′,α,ν)= 0 and α′ = α + 1 (ν = 0)
      )
   )
)
```

If there is a witnessing sequence for β, then this search will find it. The only stack value which witnesses β being false is $2^{\beta+1} + 2^{\beta}$ if β is a limit, and if β

is a successor, then $2^{\beta+1} + 2^{\beta} + \tau$, where τ witnesses the truth value $T(\beta - 1)$, and $W(\beta, \beta - 1, T(\beta - 1)) = 0$.

Theorem 5.2.1: `Truth(`β`)` computes the truth value of the sentence in the language $\{\in\}$ with constant parameters which β codes.

Proof: We reduce truth in ZFC with parameters to a computation of the recursive truth predicate for the constructible universe, as in ([6], section 6). Then we write an abstract ordinal program to compute the syntactic operations, as in lemma 5.1.3, to satisfy definition 5.2.5. As we explained before and after definition 17, a proof of $T(\beta)$ is contained in a witnessing sequence $\beta > \beta_1 > \cdots beta_i > \cdots \beta_{n-1} = \beta_n$. If `Stack` codes a witnessing sequence with the coding described in definition 11, then `Truth(`β`)` will halt and return the truth value of β, for in the computation of `Truth(`β`)`, the pair $(\alpha' + 1, \alpha)$ become the least two exponents of the stack. If α is a limit and $\alpha' = \alpha$, then the while loop repeatedly sets $(\alpha'\alpha)$ equal to each pair (β_i, β_{i+1}) of stack elements and checks that $T(\beta_i) = 1 = W(\beta_i, \beta_{i+1}, T(\beta_i))$ or $T(\beta_i) = 0 = W(\beta_i, \beta_{i+1}, T(\beta_i))$ and $\beta_i = \beta_{i+1} + 1$. We need to know that `Stack` will eventually code the witnessing sequence for β. But focusing on how `Push(Stack, ` $\cdots$`)` is called in the program, we see that `Stack` will eventually code every decreasing sequence of ordinals $\beta > \beta_1 > \cdots beta_i > \cdots \beta_{n-1} \geq \beta_n$ for which $\beta_{i+1} \leq$ the least witness for $T(\beta_i)$. □.

5.3 How many registers are necessary in a universal ordinal register machine?

Consider, first, ordinary register machines storing natural numbers.

Definition 5.3.1: A register machine has the following three commands

- `Zero(`x`)` : erases the value of register x.
- x `++` : increments the value of register x.
- `if` $x = y$ `goto` i `else` j : a general switch.

A **For** program uses `goto` loops only to model the commands

- `for` x `from` 0 `to` z `(loop)`.
- `if` $(x = y)$ `(instructions)`.

A **While** program lacks `Zero(`x`)` and `goto`, but has the commands

- x `--` : decrements the value of register x.
- `while(`$x > 0$`;` x`--) loop`.

Theorem 5.3.1: ([7] p. 205) 5-variable `While`-programs simulate Turing machines.

The proof is by storing the bit strings on the Turing tape left and right of the active head as register values. When the active head goes right, the bit string to the left increases by $2\times$, and the bit string to the right decreases by $1/2$.

Theorem 5.3.2: ([7] pp. 255-8) `While`-programs using 2 variables can simulate all `While`-programs. `FOR`-programs using 3 variables can simulate all `While`-programs.

The proof is by storing all the registers as $2^{x_0} \times 3^{x_1} \times ...{p_n}^{x_n}$, then copying these values to another register, and meanwhile altering or comparing them according to how the many-register program would have altered or compared them in its active command.

Definition 5.3.2: Let OC^n be the set of n-register well-structured ordinal computer programs (as in definition 2). Say $\rho : Ord^n \to \alpha^n$ reflects OC^n if for each P in OC^n, the function f_P which takes the inputs to P to the output of P, commutes with ρ. Let L_n be the vocabulary with a function for each n-register program: $L_n = \{Ord, <, =\} \cup \{f_P : P \in OC^n\}$, and let $FO^k(L)$ be the first order formulas in the language L, to quantifier depth k.

Definition 5.3.3: Let ρ_0 be the function $\rho_0(\alpha) = \alpha \mathrm{mod} \omega$.

Let ρ_1 be the identity below ω, and be $\omega + \rho_0$ above ω.

Let ρ_2 be the identity below $\omega \times 2$, and be $\omega \times 2 + \rho_0$ above $\omega \times 2$.

Let $\rho_3(\alpha) = \alpha \mathrm{mod} \omega^\omega$.

Let ρ_4 be the identity below ω^ω, and be $\omega^\omega + \rho_3$ above ω^ω.

Let $\rho_5(\alpha, \beta)$ be the pair $(\rho_4(\alpha), \rho_4(\alpha) + \rho_4(\beta - \alpha))$ if $\alpha \leq \beta$ and be undefined if $\alpha > \beta$.

Lemma 5.3.1: $\rho_1 : Ord \to \omega \times 2$ reflects OC^1, is the minimal reflection preserving $FO^1(L_1)$, and preserves even $FO^2(L_1)$. ρ_2 preserves $FO^3(L_1)$.

Proof: In a well-structured program with only 1 variable, `for` a `to` a `(`L`)` never executes its loop, and `if` $a = a$ `(`L`)` always executes its instructions. The result of the computation, on input a, is $a + n_P$ or n_P, depending on whether the instruction `Zero`(a) occurs and executes. It is easy to check that $\rho(P(a)) = P(\rho(a))$. If $\forall a P(a) \neq Q(a)$, then, as P and Q are constants or linear functions, we get four cases, in all of which $\forall a \rho(P(a)) \neq \rho(Q(a))$, and similarly for $\forall a P(a) < Q(a)$ and other atomic relationships replacing $\neq$, we can check the language's preservation. □

Lemma 5.3.2: $\rho_5 : Ord^2 \to (\omega^\omega \times 3)^2$ reflects OC^2, is minimal such that it preserves $FO^2(L_2)$, and preserves $FO(L_2)$.

Proof (that ρ_5 is minimal): If $L_{\times\omega}$ `= Zero(`b`); for`b `to` a `(`$a++$`).`, then $L_{\times\omega}(a, b) = (a \times \omega, a \times \omega)$. (Proof: Let a initially be a_0. When b reaches $a_0 \times n$, a reaches $a_0 \times (n + 1)$. □) If $L_{\times\omega^n} = L_{\times\omega}$ repeated n times, then $L_{\times\omega^n}(a, b) = (a \times \omega^n, a \times \omega^n)$; If L_p `= Zero(`b`); for` b `to` a `(for` b `to` a `(`$a++$`);` $a++$`)`, then $L_p(a, b) = (a \times \omega + \omega^2, a \times \omega + \omega^2)$. (Proof: The first run through the inner loop produces $(a \times \omega + 1, a \times \omega)$. Further runs through the inner loop produce $(a \times \omega + \omega \times n + 1, a \times \omega + \omega \times n)$, which are finally equal at $(a \times \omega + \omega^2, a \times \omega + \omega^2)$. In this way, we can generate L_q for any linear (in a) polynomial (in ω) $q(a, \omega)$ we wish to see as the output. □ Proof (reflection) First, observe that $P \in OC^2$ is equivalent to a program $P' \in OC^2$ which is only one loop deep.

For instance, we can write a two-loop-deep program to produce the value ω^2: `Zero(`x`); for` x `to` y `(for` x `to` y `(`y`++); `y`++)` takes any finite input

to ω^2, just the same as running y up to ω and then running x up to y. Similarly, `Zero` x`; for` x `to` y `(for` x `to` y`(for` x `to` y`(`y`++); `y`++); `y`++)` takes any finite input to ω^3, just the same limit as running y up to ω, then running x up to y, then running y up to x. The proof relies on the rule in definition 2 which prevents a loop index or bound from being erased. As a result, the order between them is fixed, and can only be made to fail during the loop by incrementing the index. Then, this finite difference can be exploited by an interior `for` loop. However, the variables could be imagined to be switched, then, so that the order relation "index < bound" can be imagined to be strict throughout the whole operation of the main loop. In this case, repeatedly chasing the bound only results in finding the next "limit of f-closed ordinals," and ω^n provide infinitely many limits of limits of... f-closed ordinals, where f is any function that can be produced within an interior loop. Those same functions can be computed, then by a sequence of loops without inner loops, which push the loop bound high enough, and then run the index up to it.

As was observed in the run of L_p now happens generally: after the inner loop has run, $a = b$. Subsequent operations inside the outer loop can only make b finitely larger than a. Second, inside any loop, the loop index grows at least linearly in time, and the loop bound grows at most linearly in time. To "Zero" the index or bound of a loop, in the loop, is illegal by definition 2, so if a loop is called, the order relation between the variables is fixed (up to a finite amount) throughout. An interior `for` loop forces the loop and index to be the same, and `if` has no effect on the values. So for any $P \in OC^2$, P is bound by a function $q(a, \omega)$, linear (in a), polynomial (in ω). □.

Lemma 5.3.3: Ord^3 reflects below $\epsilon_{\omega\times 4}$, and not lower.

Proof (not lower): The program $y++$`; for` x `to` y `(Zero(`z`); for` z `to` y `(`$y++$`))` halts at the first ϵ-number (closed under $\alpha \to \omega^\alpha$) above the initial value of y. Repeating the loop n-many times finds the n-th ϵ-number above the initial value of y. □ Proof(reflection): Suppose the first loop is $L_0 =$ `for(`$x = a; x < y; x++$`),` where a can be x, z, or 0 (same as `Zero(`x`); for(`$x =$ x`...)`. This same loop format can be repeated, as in `for(`$x = 0; x < y; x++$`) (for(`$x = x; x < y; x++$`) (`L`); `$y++$`)`. Let $f(x, y, z)$ be the supremum of the register values after applying the loop L to the initial register values x, y, z. The inner loop ends when x reaches an ordinal γ which is closed under f (γ is f-closed if $f(x, y, z) < \gamma$ whenever $x, y, z < \gamma$). Then in the outer loop, y increments, and so we reach γ_1 f-closed, and so on. The outer loop ends at the first γ which is a limit of f-closed ordinals. Now $x < y$ is fixed for the duration of the computation. For if x were incremented above y infinitely often, then y is also incremented above x infinitely often (before each consideration of the bounding clause $x < y$), so that at time $\sup t_i$, where t_{2i+1} is the next time x is larger, and t_{2i+2} is the next time y is larger, then $x = y$ again, and as this is a limit time, we are checking the bounding clause $x < y$, and the loop ends. So if x exceeds y infinitely often, then the loop ends. So, without loss of generality, suppose $x < y$ always holds, and consider what an inner loop can do. Incrementing x is counter-productive, since it hastens the time when $x = y$ will be attained. Incrementing y is a great idea, but the only clock available is `Zero(`z`); for` z `to` y`(`$f(y)$`)`, which executes until $z + \alpha$, incrementing once each loop, reaches $f^\alpha(y)$, the α-th iteration of f, whatever function is in the

innermost part. This function could, at most, be `for` z `to` x or `for` x `to` z, in which cases $f(y)$ would increase y some infinite number of times, but never more than its own value, so $f(y) < y + y$.□

If the initial register values are 0, then we cannot compute anything beyond ϵ_ω. But if the initial register values are given, then we reflect the first one into $(\epsilon_\omega, \epsilon_{\omega\times 2})$ and the next into $(\epsilon_{\omega\times 2}, \epsilon_{\omega\times 3})$, the third into $(\epsilon_{\omega\times 3}, \epsilon_{\omega\times 4})$, and the same proof shows that subsequent computation stays below $\epsilon_{\omega\times 4}$.

Theorem 5.3.3: An ordinal computer with fewer than four registers cannot be a universal ordinal computer. However, an ordinal computer with ten registers can model a universal ordinal computer.

Proof: We have proven in the lemmas that fewer than four registers is insufficient, since these computers reflect below small ordinals. The program in definition 5.2.7 is written using ten registers. That is, it uses the five variables β, `Stack`, i, α, α', (we can recompute the loop limit 2^β each time we check $i < 2^\beta$) and then calls `Pop`, which uses as local variables a `Small Stack`, a `Temp Stack`, ϵ to search between 0 and $\beta + 1$, γ, and a fifth register, which might sometimes store the sum $\alpha + 2^\epsilon +$ `SmallStack`, and sometimes be the loop index κ in the last line of `Pop`. The register for `Pop`'s γ, which we could call `Large Stack` in analogy with `Small Stack` never exceeds `Stack`; we can make it larger than `Stack` when it's time for the `while` loop in definition 5.2.7 to halt. If γ can code the bit of information that halts the `while` loop, then that loop doesn't need a register dedicated to indexing it. □

We would like to indicate how four registers are sufficient for a universal program on an ordinal register machine. We simulate an n-register machine by putting all n variables onto two stacks. We copy the information from one stack to the other, and change the appropriate i-th register in the process, as in the proof of theorem 5.3.2. The fourth variable contains the value of a single element. When that element is erased on the stack, we copy its value more deeply into the stack, where it won't be erased by the varying and limiting of values lower on the stack. We do not have a clear and convincing proof of this.

Conjecture 5.3.1: Four registers suffice for a universal program on an ordinal register machine.

5.4 Complexity

For ordinal register machines, it is possible to compare the runtime of a program to its input values, and therefore it is reasonable to talk about the bounds on the complexity of problems for such machines.

Our program for computing truth (definition 5.2.7) runs in time at most ordinal-exponential in the input β. A similar program, described in [5], runs in ordinal-polynomial time: to determine the truth predicate, when ordinally many bit registers are available, search the registers below α to find a witness for α. This takes time $\sum\{\beta : \beta < \alpha\}$ If $\alpha = \omega^\gamma$ for some γ, this is α. In any case, the sum is $< \alpha^2$. This program runs faster than definition 5.2.7 because it can store the whole recursive truth predicate up to β when computing $F(\beta)$. It seems intuitively clear that a computer with finitely many ordinal registers cannot run in time faster than $O(2^\beta)$, i.e., that it must compute $F(\beta_1)...F(\beta_n)$ for every finite sequence $\{\beta_i : i < n\}$ of ordinals $< \beta$.

References

[1] R. Bissell-Siders, *Ordinal computers.* math.LO/9804076 at arXiv.org, 1998.

[2] H. Friedman, *Algorithmic procedures, generalized Turing algorithms, and elementary recursion theory.* Logic Colloquium '69 (Proc. Summer School and Colloq., Manchester, 1969), pp. 361–389. North-Holland, Amsterdam, 1971.

[3] G. Jacopini and C. Böhm, *Flow Diagrams, Turing Machines, and Languages with Only Two Formation Rules.* Comm ACM, 9,5 May 1966.

[4] J. Hamkins and A. Lewis, *Infinite Time Turing Machines.* J. Symbolic Logic, 65(2): 567-604, 2000.

[5] P. Koepke, *Turing Computations on Ordinals.* Bulletin of Symbolic Logic, 11(3): 377-397, 2005.

[6] P. Koepke and R. Siders, *Register Computations on Ordinals.* submitted Feb 2006. For the moment, see

http://www.math.helsinki.fi/∼rsiders/Papers/RegistersOnOrdinals/

[7] M. Minsky, *Computation: Finite and Infinite Machines.* Prentice-Hall, 1967.

[8] J. Shepherdson and H. Sturgis, *Computability of recursive functions*, J. Assoc. Comput. Mach. 10 217–255, 1963.

[9] J. Tucker and J. Zucker, *Computable functions and semicomputable sets on many sorted algebras*, in S. Abramsky, D. Gabbay and T Maibaum (eds.) *Handbook of Logic for Computer Science*, Volume V, Oxford University Press, 317-523.

6. REGISTER COMPUTATIONS ON ORDINALS

with Professor Peter Koepke, Mathematisches Institut, Universität Bonn.

abstract

We generalize ordinary register machines on natural numbers to machines whose registers contain arbitrary ordinals. *Ordinal register machines* are able to compute a recursive bounded truth predicate on the ordinals. The class of sets of ordinals which can be read off the truth predicate satisfies a natural theory SO. SO is the theory of the sets of ordinals in a model of the ZERMELO-FRAENKEL axioms ZFC. This allows the following characterization of computable sets: a set of ordinals is ordinal register computable if and only if it is an element of GÖDEL's constructible universe L.

Introduction.

There are many equivalent machine models for defining the class of intuitively computable sets. We shall model computations on ordinals on the *unlimited register machines* (*URM*) presented in [2]. An URM has registers $R_0, R_1, \ldots$ which can hold *natural numbers*, i.e., elements of the set $\omega = \{0, 1, \ldots\}$. A register program consists of commands to reset, increase, or copy a register. The program may jump on condition of equality between two registers. An obvious generalization from the perspective of transfinite ordinal theory is to extend such calculations to the class $\mathrm{Ord} = \{0, 1, \ldots, \omega, \omega+1, \ldots\}$ of all *ordinal numbers* so that registers may contain arbitrary ordinals. At *limit* ordinals one defines the program states and the registers contents by appropriate limit operations.

This notion of *ordinal (register) computability* obviously extends standard register computability. By the CHURCH-TURING thesis recursive operations on natural numbers are ordinal computable. The ordinal arithmetic operations (addition, multiplication, exponentiation) and GÖDEL's pairing function $G : \mathrm{Ord} \times \mathrm{Ord} \to \mathrm{Ord}$ are also ordinal computable.

Using the pairing function one can interpret each ordinal α as a first-order sentence with constant symbols for ordinals $< \alpha$. One can then define a recursive truth predicate $T \subseteq \mathrm{Ord}$ by:

$$T(\alpha) \text{ iff } (\alpha, <, G \cap \alpha^3, T \cap \alpha) \vDash \alpha.$$

This recursion can be carried out on an ordinal register machine, using stacks which contain finite decreasing sequences of ordinals. For ordinals μ and ν the

function T codes the set

$$X(\mu, \alpha) = \{\beta < \mu | T(G(\alpha, \beta))\}.$$

The class

$$\mathcal{S} = \{X(\mu, \alpha) | \mu, \alpha \in \mathrm{Ord}\}$$

is the class of sets of ordinals of a transitive proper class model of set theory. Since ordinal computations can be carried out in the $\subseteq$-smallest such model, namely GÖDEL's model L of *constructible sets*, we can characterize ordinal computability:

Theorem 6.0.1: A set $x \subseteq \mathrm{Ord}$ is ordinal computable if and only if $x \in L$.

This theorem may be viewed as an analogue of the CHURCH-TURING thesis: ordinal computability defines a natural and absolute class of sets, and it is stable with respect to technical variations in its definition. Register machines on ordinals were first considered by the second author [1]; the results proved in the present article were guided by the related *ordinal* TURING *machines* [7] which generalize the infinite-time TURING machines of [5].

6.1 Ordinal register machines

Ordinal register machines (ORM's) basically use the same instructions and programs as the *unlimited register machines* of the standard textbook by N. CUTLAND [2].

Definition 6.1.1: An ORM *program* is a finite list $P = P_0, P_1, \ldots, P_{k-1}$ of *instructions* acting of *registers* $R_0, R_1, \ldots$. The index i of the instruction P_i is also called the *state* of P_i. An instruction may be of one of four kinds:

a) the *zero instruction* `Z(n)` changes the contents of R_n to 0, leaving all other registers unaltered;

b) the *successor instruction* `S(n)` increases the ordinal contained in R_n, leaving all other registers unaltered;

c) the *transfer instruction* `T(m,n)` sets the contents of R_n to the contents of R_m, leaving all other registers unaltered;

d) the *jump instruction* P_i =`J(m,n,q)` is carried out within the program P as follows: the contents r_m and r_n of the registers R_m and R_n are compared, all registers are left unaltered; then, if $r_m = r_n$, the ORM proceeds to the instruction P_q of P; if $r_m \neq r_n$, the ORM proceeds to the next instruction P_{i+1} in P.

ORM programs are carried out along an ordinal timeline. At each ordinal time t the machine will be in a *configuration* consisting of a program state $I(t) \in \omega$ and register contents which can be viewed as a function $R(t) : \omega \to \mathrm{Ord}$. $R(t)(n)$ is the content of the register R_n at time t. We also write $R_n(t)$ instead of $R(t)(n)$. The machine configuration at limit times t will be defined via inferior limits where

$$\liminf_{s \to t} \alpha_s = \bigcup_{s<t} \bigcap_{s<r<t} \alpha_r.$$

Definition 6.1.2: Let $P = P_0, P_1, \ldots, P_{k-1}$ be an ORM program. A pair

$$I : \theta \to \omega, R : \theta \to {}^{\omega}\,\mathrm{Ord}$$

is an *ordinal (register) computation* by P if the following hold:

a) θ is a successor ordinal or $\theta = \mathrm{Ord}$; θ is the *length* of the computation;

b) $I(0) = 0$; the machine starts in state 0;

c) If $t < \theta$ and $I(t) \notin k = \{0, 1, \ldots, k-1\}$ then $\theta = t + 1$; the machine *stops* if the machine state is not a program state of P;

d) If $t < \theta$ and $I(t) \in \{0, 1, \ldots, k-1\}$ then $t + 1 < \theta$; the next configuration is determined by the instruction $P_{S(t)}$:

   i. if $P_{S(t)}$ is the zero instruction `Z(n)` then let $I(t+1) = I(t)+1$ and define $R(t+1) : \omega \to \mathrm{Ord}$ by

$$R_k(t+1) = \begin{cases} 0, \text{ if } k = n \\ R_k(t), \text{ if } k \neq n \end{cases}$$

   ii. if $P_{S(t)}$ is the successor instruction `S(n)` then let $I(t+1) = I(t)+1$ and define $R(t+1) : \omega \to \mathrm{Ord}$ by

$$R_k(t+1) = \begin{cases} R_k(t)+1, \text{ if } k = n \\ R_k(t), \text{ if } k \neq n \end{cases}$$

   iii. if $P_{S(t)}$ is the transfer instruction `T(m,n)` then let $I(t+1) = I(t)+1$ and define $R(t+1) : \omega \to \mathrm{Ord}$ by

$$R_k(t+1) = \begin{cases} R_m(t), \text{ if } k = n \\ R_k(t), \text{ if } k \neq n \end{cases}$$

   iv. if $P_{S(t)}$ is the jump instruction `J(m,n,q)` then let $R(t+1) = R(t)$ and

$$I(t+1) = \begin{cases} q, \text{ if } R_m(t) = R_n(t) \\ I(t)+1, \text{ if } R_m(t) \neq R_n(t) \end{cases}$$

e) If $t < \theta$ is a limit ordinal, the machine constellation at t is determined by taking inferior limits:

$$\begin{aligned} \forall k \in \omega \; R_k(t) &= \liminf_{r \to t} R_k(r); \\ I(t) &= \liminf_{r \to t} I(r). \end{aligned}$$

The ordinal computation is obviously recursively determined by the initial register contents $R(0)$ and the program P. We call it the *ordinal computation by P with imput $R(0)$*. If the computation stops, $\theta = \beta + 1$ is a successor ordinal and $R(\beta)$ is the final register content. In this case we say that P *computes* $R(\beta)(0)$ from $R(0)$ and write $P : R(0) \mapsto R(\beta)(0)$.

The definition of the state $I(t)$ for limit t can be motivated as follows. Since a program is finite its execution will lead to some (complex) looping structure involving loops, subloops and so forth. This can be presented by pseudo code like:

```
      ...
17:begin loop
          ...
    21:   begin subloop
             ...
    29:   end subloop
          ...
32:end loop
      ...
```

Assume that for times $r \to t$ the loop $(17 - 32)$ with its subloop $(21 - 29)$ is traversed cofinally often. Then at time t it seems natural to put the machine at the start of the "main loop". Assuming that the lines of the program are enumerated in increasing order this corresponds to the lim inf rule

$$I(t) = \liminf_{r \to t} I(r).$$

The interpretation of programs by computations yields associated notions of computability.

Definition 6.1.3: An n-ary partial function $F : \mathrm{Ord}^m \rightharpoonup \mathrm{Ord}$ is *ordinal (register) computable* if there are a register program P and ordinals $\delta_0, \ldots, \delta_{n-1}$ such that for every m-tuple $(\alpha_0, \ldots, \alpha_{m-1}) \in \mathrm{dom}\, F$ holds

$$P : (\alpha_0, \ldots, \alpha_{m-1}, \delta_0, \ldots, \delta_{n-1}, 0, 0, \ldots) \mapsto F(\alpha_0, \ldots, \alpha_{m-1}).$$

A subset $x \subseteq \mathrm{Ord}$ is *ordinal (register) computable* if its characteristic function χ_x is ordinal computable.

6.2 Algorithms

Since ordinal register machines are a straightforward extension of standard register machines, all recursive functions can be computed by an ordinal register machine. We shall now show that basic operations on ordinal numbers are ordinal register computable. We present programs in an informal *pseudo code* where variables correspond to registers.

```
Ordinal addition, computing gamma = alpha + beta:
0  alpha':=0
1  beta':=0
2  gamma:=0
3  if alpha=alpha' then go to 7
4  alpha':=alpha'+1
5  gamma:=gamma+1
6  go to 3
7  if beta=beta' then STOP
8  beta':=beta'+1
9  gamma:=gamma+1
10 go to 7
```

Observe that at limit times this algorithm, by the lim inf rule, will nicely cycle back to the beginnings of loops `3 - 6` or `7 - 10` resp.

```
Ordinal multiplication, computing gamma = alpha * beta:
0  beta':=0
1  gamma:=0
2  if beta=beta' then STOP
3  beta':=beta'+1
4  gamma:=gamma + alpha
5  go to 2
```

We interpret the program line `gamma:=gamma + alpha` as a *macro*, i.e., the above addition program has to be substituted for that line with reasonable modifications of variables, registers and line numbers. Also adequate transfer of arguments and values between variables has to be arranged.

In general this substitution technique yields the closure under composition for the class of ordinal computable functions:

Theorem 6.2.1: Let $f(v_0, \ldots, v_{n-1})$ and $g_0(\overrightarrow{w}), \ldots, g_{n-1}(\overrightarrow{w})$ be ordinal computable functions. Then the composition $h(\overrightarrow{w}) = f(g_0(\overrightarrow{w}), \ldots, g_{n-1}(\overrightarrow{w}))$ is ordinal computable.

The GÖDEL pairing function for ordinals is important for coding information into single ordinals. It is defined recursively by

$$\begin{aligned} G(\alpha,\beta) \quad = \quad \{G(\alpha',\beta') \mid & \max(\alpha',\beta') < \max(\alpha,\beta) \text{ or} \\ & (\max(\alpha',\beta') = \max(\alpha,\beta) \text{ and } \alpha' < \alpha) \text{ or} \\ & (\max(\alpha',\beta') = \max(\alpha,\beta) \text{ and } \alpha' = \alpha \text{ and } \beta' < \beta)\}. \end{aligned}$$

We sketch an algorithm for computing $\gamma = G(\alpha, \beta)$, it proceeds by increasing a pair (α', β') along the well-order of $\mathrm{Ord} \times \mathrm{Ord}$ implicit in the definition of G until (α, β) is reached and simultaneously increasing the ordinal γ along the ordinals.

```
Goedel pairing, computing gamma = G(alpha,beta):
0  alpha':=0
1  beta':=0
2  eta:=0
3  flag:=0
4  gamma:=0
5  if alpha=alpha' and beta=beta' then STOP
6  if alpha'=eta and and beta'=eta and flag=0 then
    alpha':=0, flag:=1, gamma:=gamma+1, go to 5 fi
7  if alpha'=eta and and beta'=eta and flag=1 then
    eta:=eta+1,alpha':=eta,beta':=0,gamma:=gamma+1, go to 5 fi
8  if beta'<eta and flag=0 then
    beta':=beta'+1, gamma:=gamma+1, go to 5 fi
9  if alpha'<eta and flag=1 then
    alpha':=alpha'+1, gamma:=gamma+1, go to 5 fi
```

The inverse functions G_0 and G_1 satisfying

$$\forall \gamma \gamma = G(G_0(\gamma), G_1(\gamma))$$

are also ordinal computable: compute $G(\alpha, \beta)$ for $\alpha, \beta < \gamma$ until you find α, β with $G(\alpha, \beta) = \gamma$; then set $G_0(\gamma) = \alpha$ and $G_1(\gamma) = \beta$. This is a special case of the following inverse function theorem.

Theorem 6.2.2: Let the function $f : \mathrm{Ord}^n \to \mathrm{Ord}$ be ordinal computable and surjective. Then there are ordinal computable functions $g_0, \ldots, g_{n-1} : \mathrm{Ord} \to \mathrm{Ord}$ such that

$$\forall \alpha f(g_0(\alpha), \ldots, g_{n-1}(\alpha)) = \alpha.$$

6.3 3-adic representations and ordinal stacks

We shall compute a recursive truth function using a *stack* that can hold a (finite) sequence $\alpha_0 > \alpha_1 > \ldots > \alpha_{n-2} \geqslant \alpha_{n-1}$ of ordinals which is strictly decreasing except possibly for the last two ordinals. This sequence of ordinals will be coded into a single ordinal by 3-adic representations.

Proposition 6.3.1: Let $\delta > 1$ be a fixed *basis* ordinal. A representation

$$\alpha = \delta^{\alpha_0} \cdot \zeta_0 + \delta^{\alpha_1} \cdot \zeta_1 + \ldots + \delta^{\alpha_{n-1}} \cdot \zeta_{n-1}$$

with $\alpha_0 > \alpha_1 > \ldots > \alpha_{n-1}$ and $0 < \zeta_0, \zeta_1, \ldots, \zeta_{n-1} < \delta$ is called a δ-*adic representation* of α.

It is an easy exercise in ordinal arithmetic to show that every $\alpha \in \mathrm{Ord}$ possesses a unique δ-adic representation. So a decreasing stack $\alpha_0 > \alpha_1 > \ldots > \alpha_{n-2} \geqslant \alpha_{n-1}$ of ordinals can be coded by

$$\alpha = \langle \alpha_0, \alpha_1, \ldots, \alpha_{n-2}, \alpha_{n-1} \rangle = 3^{\alpha_0} + 3^{\alpha_1} + \ldots + 3^{\alpha_{n-2}} + 3^{\alpha_{n-1}}.$$

We call the natural number n the *length* of the stack α. The elements α_{n-1}, α_{n-2}, ... of this stack can be defined from α as follows:

$$\begin{aligned} \alpha_{n-1} &= \text{the largest } \xi \text{ such that there is } \zeta \text{ with } \alpha = 3^\xi \cdot \zeta \\ \alpha_{n-2} &= \text{the largest } \xi \text{ such that there is } \zeta \text{ with } \alpha = 3^\xi \cdot \zeta + 3^{\alpha_{n-1}} \\ &\ldots \end{aligned}$$

Since the ordinal arithmetic operations are ordinal computable, the ordinals $\alpha_{n-1}, \alpha_{n-2}$ are ordinal computable by some programs `last`, `llast` resp. We assume that these functions return a special value `UNDEFINED` if the stack is too short.

The computation in the subsequent recursion theorem proceeds by ranging over previous arguments and values in a systematic way. This can be organized by a stack, due to the limit behaviour of stacks.

Proposition 6.3.2: Let $t \in \mathrm{Ord}$ be a limit time and $t_0 < t$. For time $\tau \in [t_0, t)$ let the contents of the stack register `stack` be of the form

$$\alpha_\tau = \langle \alpha_0, \ldots, \alpha_{k-1}\ , \rho(\tau), \ldots \rangle$$

with fixed $\alpha_0, \ldots, \alpha_{k-1}$ and variable $\rho(\tau) \leqslant \alpha_{k-1}$. Assume that the sequence $(\rho(\tau) | \tau \in [t_0, t))$ is weakly monotonously increasing and that the length of `stack` is equal to $k+1$ cofinally often below t. Then at limit time t the content of `stack` is of the form

$$\alpha_t = \langle \alpha_0, \ldots, \alpha_{k-1}\ , \rho \rangle$$

with $\rho = \bigcup_{\tau \in [t_0, t)} \rho(\tau)$.

6.4 A recursion theorem

Theorem 6.4.1: Let $H : \mathrm{Ord}^3 \to \mathrm{Ord}$ be ordinal computable and define $F : \mathrm{Ord} \to \mathrm{Ord}$ recursively by

$$F(\alpha) = \begin{cases} 1 \text{ iff } \exists \nu < \alpha \; H(\alpha, \nu, F(\nu)) = 1 \\ 0 \text{ else} \end{cases}$$

Then F is ordinal computable.

Given an algorithm for the recursion function H we compute F with a `stack` as considered above and a register `value` which can hold a single value of the function F: we let `value = 2` stand for "undefined'. The following program P accepts an input ordinal α on the singleton stack $\langle \alpha \rangle$ and stops with the output stack $\langle \alpha \rangle$ and `value`$= F(\alpha)$. During the recursion the program will call itself with non-empty stacks $\alpha = \alpha_0, \alpha_1, \ldots, \alpha_{n-1}$ and compute the value $F(\alpha_{n-1})$. The main loop of the program serves to let the bounded quantifier $\exists \nu < \alpha$ range over all $\nu < \alpha$. The subloop evaluates the kernel $H(\alpha, \nu, F(\nu)) = 1$ of the quantifier and returns the result for further calculation of values.

```
  value:=2                %% set value to undefined
MainLoop:
  nu:=last(stack)
  alpha:=llast(stack)
  if nu = alpha then
1:  do
    remove_last_element_of(stack)
    value:=0              %% set value equal to 0
    goto SubLoop
    end
  else
2:  do
    stack:=stack + 1      %% push the ordinal 0 onto the stack
    goto MainLoop
    end
SubLoop:
  nu:=last(stack)
  alpha:=llast(stack)
  if alpha = UNDEFINED then STOP
  else
    do
    if H(alpha,nu,value)=1 then
3:    do
      remove_last_element_of(stack)
      value:=1
      goto SubLoop
      end
    else
4:    do
      stack:=stack + (3**y)*2  %% push y+1
      value:=2                 %% set value to undefined
```

```
    goto MainLoop
    end
  end
```

The correctness of the program is established by

Theorem 6.4.2: The ordinal computation I, R by the program P has the following properties

a) If I, R is in state `MainLoop` at time s with `stack` contents $\langle \alpha_0, \ldots, \alpha_{n-1} \rangle$ where $n \geqslant 1$ then I, R will get into state `SubLoop` at a later time t with the same `stack` contents $\langle \alpha_0, \ldots, \alpha_{n-1} \rangle$ and the register `value` holding the value $F(\alpha_{n-1})$. Moreover in the interval $[s, t)$ the contents of `stack` will always be at least as big as $\langle \alpha_0, \ldots, \alpha_{n-1} \rangle$.

b) Let I, R be in state `MainLoop` at time s with `stack` contents $\alpha_0 > \ldots > \alpha_{n-1}$ where $n \geqslant 1$. Define $\bar{\alpha}$ = the minimal ordinal $\nu < \alpha_{n-1}$ such that $H(\alpha_{n-1}, \nu, F(\nu)) = 1$ if this exists and $\bar{\alpha} = \alpha_{n-1}$ else. Then there is a strictly increasing sequence $(t_i | i \leqslant \bar{\alpha})$ of times $t_i > t$ such that I, R is in state `MainLoop` at time t_i with `stack` contents $\langle \alpha_0, \ldots, \alpha_{n-1}, i \rangle$. Moreover in every time interval $[t_i, t_{i+1})$ the `stack` contents are $\geqslant \langle \alpha_0, \ldots, \alpha_{n-1}, i \rangle$.

c) If I, R is in state `MainLoop` with `stack` contents $\langle \alpha \rangle$ then it will later *stop* with `stack` contents $\langle \alpha \rangle$ and the register `value` holding the value $F(\alpha)$. Hence the function F is ordinal register computable.

Proof a) and b) are proved simultaneously by induction over the last element α_{n-1} of the stack. Assume that P is in state `MainLoop` at time s with stack contents $\langle \alpha_0, \ldots, \alpha_{n-1} \rangle$ where $n \geqslant 1$ and that a) and b) hold for all stack contents $\langle \beta_0, \beta_1, \ldots, \beta_{m-1} \rangle$ with $\beta_{m-1} < \alpha_{n-1}$. Define $\bar{\alpha}$ as in b).

We first prove b) by defining an appropriate sequence $(t_i | i \leqslant \bar{\alpha})$ by recursion over $i \leqslant \bar{\alpha}$.
$i = 0$. Inspection of P shows that the computation will move to state `2` and obtain `stack` contents $\langle \alpha_0, \ldots, \alpha_{n-1}, 0 \rangle$ before immediately returning to `MainLoop`.
$i = j+1$ where $j < \bar{\alpha}$. By recursion, P is in state `MainLoop` at time t_j with stack contents $\langle \alpha_0, \ldots, \alpha_{n-1}, j \rangle$. $j < \bar{\alpha} \leqslant \alpha_{n-1}$ so that the inductive assumption a) holds for $\langle \alpha_0, \ldots, \alpha_{n-1}, j \rangle$. So there will be a later time when P is in state `SubLoop` with stack contents $\langle \alpha_0, \ldots, \alpha_{n-1}, j \rangle$ and `value`$= F(j)$. Also during that computation the stack contents will always be $\geqslant \langle \alpha_0, \ldots, \alpha_{n-1}, j \rangle$. Inspection of the program shows that it will further compute $H(\alpha_{n-1}, j, F(j))$. This value will be $\neq 1$ by definition of $\bar{\alpha}$. So the computation will move on to state `4` with stack contents $\langle \alpha_0, \ldots, \alpha_{n-1}, j+1 \rangle$. At the subsequent time $t_i = t_{j+1}$ the computation is in state `MainLoop` with stack contents $\langle \alpha_0, \ldots, \alpha_{n-1}, i \rangle$. i is a limit ordinal. Then by the limit behaviour of the machine and in particular by the above proposition, at time $t_i = \bigcup \{t_j | j < i\}$ the machine will be in state `MainLoop` with `stack` contents $\langle \alpha_0, \ldots, \alpha_{n-1}, i \rangle$.

Now we prove a).
Case 1: $\bar{\alpha} < \alpha_{n-1}$. Then $F(\bar{\alpha}) = 1$. By b) the computation will get to state `MainLoop` with `stack` contents $\langle \alpha_0, \ldots, \alpha_{n-1}, \bar{\alpha} \rangle$. By the inductive hypothesis, the machine will then get to state `SubLoop` with `stack` contents $\langle \alpha_0, \ldots, \alpha_{n-1}, \bar{\alpha} \rangle$ and `value` equal to $F(\bar{\alpha})$. Then the program will compute $H(\alpha_{n-1}, \bar{\alpha}, F(\bar{\alpha})) = 1$ and move into alternative `3`. The register `value` obtains

the value $F(\alpha_{n-1}) = 1$ and the computation moves to state `SubLoop` with the last stack element removed: `stack` $= \langle \alpha_0, \ldots, \alpha_{n-1} \rangle$, as required.
Case 2: $\bar{\alpha} = \alpha_{n-1}$. Then $F(\bar{\alpha}) = 0$. By b), the computation will get to state `MainLoop` with stack contents $\langle \alpha_0, \ldots, \alpha_{n-1}, \bar{\alpha} = \alpha_{n-1} \rangle$. Inspection of the program shows that it will get into alternative `1`, set `stack:`$= \langle \alpha_0, \ldots, \alpha_{n-1} \rangle$, `value:`$= 0$ and move to `SubLoop`, which proves a) in this case.

Finally, c) follows readily from a) and inspection of the program. □

6.5 A recursive truth predicate

The ordinal arithmetic operations and the GÖDEL pairing function G allow us to code finite sequences of ordinals into single ordinals. The coding can be made ordinal computable in the sense that usual operations on finite sequences like concatenation or substitution are computable as well. This allows to code formal languages in an ordinal computable way.

We shall consider a language L_R appropriate for first-order structures of the type

$$(\alpha, <, G, R)$$

where the GÖDEL function G is viewed as a ternary *relation* on α and R is a unary relation on α. The terms of the language are variables v_n for $n < \omega$ and constant symbols c_ξ for $\xi \in \mathrm{Ord}$; the symbol c_ξ will be interpreted as the ordinal ξ. The language has atomic formulas $t_1 \equiv t_2$, $t_1 < t_2$, $\dot{G}(t_1, t_2, t_3)$ and $\dot{R}(t_1)$. The symbol $\dot{G}$ will be interpreted by the GÖDEL relation G. If φ and ψ are (compound) formulas of the language, $n < \omega$, and t is a term then

$$\neg\varphi, (\varphi \vee \psi), \text{ and } (\exists v_n < t)\ \varphi$$

are also formulas; thus we are only working with bounded quantifications. We assume an ordinal computable coding such that a bounded existential quantification $(\exists v_n < c_\xi)\ \varphi$ is coded by a larger ordinal than each of its instances $\varphi\frac{c_\zeta}{v_n}$ with $\zeta < \xi$:

$$\varphi\frac{c_\zeta}{v_n} < (\exists v_n < c_\xi)\ \varphi).$$

An L_R-formula is an *L_R-sentence* if it does not have free variables. If φ is an L_R-sentence so that all constants symbols c_ξ in φ have indices $\xi < \alpha$ then the satisfaction relation

$$(\alpha, <, G, R) \vDash \varphi$$

is defined as usual. Bounded sentences are *absolute* for sufficiently long initial segments of the ordinals. If φ is a bounded sentence such that every constant symbol c_ξ occuring in φ satisfies $\xi < \beta < \alpha$ then

$$(\alpha, <, G, R) \vDash \varphi \text{ iff } (\beta, <, G, R) \vDash \varphi.$$

We may assume that the coding of formulas by ordinals φ will satisfy that $\xi < \varphi$ for every constant symbol c_ξ occuring in φ. So the meaning of a bounded sentence φ is given by

$$(\varphi, <, G, R) \vDash \varphi.$$

This leads to the recursive definition of a *bounded truth predicate* $T \subseteq \mathrm{Ord}$ over the ordinals

$$T(\alpha) \text{ iff } \alpha \text{ is a bounded } L_R\text{-sentence and } (\alpha, <, G, T \cap \alpha) \vDash \alpha.$$

We shall see that T is a strong predicate which codes a model of set theory. We first show that the characteristic function χ_T of T can be defined according to the recursion scheme

$$\chi_T(\alpha) = \begin{cases} 1 \text{ iff } (\exists \nu < \alpha)\ H(\alpha, \nu, \chi_T(\nu)) = 1 \\ 0 \text{ else} \end{cases}$$

with an appropriate computable recursion function H.

$$\begin{aligned} H(\alpha, \nu, \chi) = 1 \quad \text{iff} \quad & \alpha \text{ is an } L_R\text{-sentence and} \\ & \exists \xi, \zeta < \alpha\ (\alpha = c_\xi \equiv c_\zeta \wedge \xi = \zeta) \\ \text{or} \quad & \exists \xi, \zeta < \alpha\ (\alpha = c_\xi < c_\zeta \wedge \xi < \zeta) \\ \text{or} \quad & \exists \xi, \zeta, \eta < \alpha\ (\alpha = \dot{G}(c_\xi, c_\zeta, c_\eta) \wedge \eta = G(\xi, \zeta)) \\ \text{or} \quad & \exists \xi < \alpha\ (\alpha = \dot{R}(c_\xi) \wedge \nu = \xi \wedge \chi = 1) \\ \text{or} \quad & \exists \varphi < \alpha\ (\alpha = \neg\varphi \wedge \nu = \varphi \wedge \chi = 0) \\ \text{or} \quad & \exists \varphi, \psi < \alpha\ (\alpha = (\varphi \vee \psi) \wedge (\nu = \varphi \vee \nu = \psi) \wedge \chi = 1) \\ \text{or} \quad & \exists n < \omega\ \exists \xi < \alpha\ \exists \varphi < \alpha \\ & \qquad (\alpha = (\exists v_n < c_\xi)\ \varphi \wedge (\exists \zeta < \xi)\ \nu = \varphi \frac{c_\zeta}{v_n} \wedge \chi = 1). \end{aligned}$$

Then χ_T and T are ordinal register computable by the recursion theorem 6.4.1.

6.6 The theory SO of sets of ordinals

It is well-known that a model of Zermelo-Fraenkel set theory with the axiom of choice (ZFC) is determined by its sets of ordinals (see [6], Theorem 13.28). We define a natural theory SO which axiomatizes the sets of ordinals in a model of ZFC. The theory SO is two-sorted: *ordinals* are taken as atomic objects, the other sort corresponds to *sets of ordinals*. Let L_{SO} be the language

$$L_{\mathrm{SO}} := \{\mathrm{Ord}, \mathrm{SOrd}, <, =, \in, g\}$$

where Ord and SOrd are unary predicate symbols, $<$, $=$ and $\in$ are binary predicate symbols and g is a two-place function. To simplify notation, we use lower case greek letters to range over elements of Ord and lower case roman letters to range over elements of SOrd.

1. Well-ordering axiom:
   $\forall \alpha, \beta, \gamma(\neg\alpha < \alpha \wedge (\alpha < \beta \wedge \beta < \gamma \to \alpha < \gamma) \wedge$
   $(\alpha < \beta \vee \alpha = \beta \vee \beta < \alpha)) \wedge$
   $\forall a(\exists \alpha(\alpha \in a) \to \exists \alpha(\alpha \in a \wedge \forall \beta(\beta < \alpha \to \neg\beta \in a)))$;

2. Axiom of infinity (existence of a limit ordinal):
   $\exists \alpha(\exists \beta(\beta < \alpha) \wedge \forall \beta(\beta < \alpha \to \exists \gamma(\beta < \gamma \wedge \gamma < \alpha)))$;

3. Axiom of extensionality: $\forall a, b(\forall \alpha(\alpha \in a \leftrightarrow \alpha \in b) \to a = b)$;

4. Initial segment axiom: $\forall\alpha\exists a\forall\beta(\beta < \alpha \leftrightarrow \beta \in a)$;

5. Boundedness axiom: $\forall a\exists\alpha\forall\beta(\beta \in a \rightarrow \beta < \alpha)$;

6. Pairing axiom (GÖDEL Pairing Function):
$\forall\alpha,\beta,\gamma(g(\beta,\gamma) \leq \alpha \leftrightarrow \forall\delta,\epsilon((\delta,\epsilon) <^* (\beta,\gamma) \rightarrow g(\delta,\epsilon) < \alpha))$.
Here $(\alpha,\beta) <^* (\gamma,\delta)$ stands for
$\exists\eta,\theta(\eta = \max(\alpha,\beta) \wedge \theta = \max(\gamma,\delta) \wedge (\eta < \theta \vee$
$(\eta = \theta \wedge \alpha < \gamma) \vee (\eta = \theta \wedge \alpha = \gamma \wedge \beta < \delta)))$,
where $\gamma = \max(\alpha,\beta)$ abbreviates $(\alpha > \beta \wedge \gamma = \alpha) \vee (\alpha \leq \beta \wedge \gamma = \beta)$;

7. g is onto: $\forall\alpha\exists\beta,\gamma\ (\alpha = g(\beta,\gamma))$;

8. Axiom schema of separation: For all L_{SO}-formulae $\phi(\alpha,P_1,\ldots,P_n)$ postulate:
$\forall P_1,\ldots,P_n\forall a\exists b\forall\alpha(\alpha \in b \leftrightarrow \alpha \in a \wedge \phi(\alpha,P_1,\ldots,P_n))$;

9. Axiom schema of replacement: For all L_{SO}-formulae $\phi(\alpha,\beta,P_1,\ldots,P_n)$ postulate:
$\forall P_1,\ldots,P_n(\forall\xi,\zeta_1,\zeta_2(\phi(\xi,\zeta_1,P_1,\ldots,P_n) \wedge \phi(\xi,\zeta_2,P_1,\ldots,P_n) \rightarrow \zeta_1 = \zeta_2)$
$\rightarrow \forall a\exists b\forall\zeta(\zeta \in b \leftrightarrow \exists\xi \in a\ \phi(\xi,\zeta,P_1,\ldots,P_n)))$;

10. Powerset axiom:
$\forall a\exists b\forall z(\exists\alpha(\alpha \in z) \wedge \forall\alpha(\alpha \in z \rightarrow \alpha \in a) \rightarrow \exists^{=1}\xi\forall\beta(\beta \in z \leftrightarrow g(\beta,\xi) \in b))$.

6.7 Assembling sets along wellfounded relations

In standard set theory a set x can be represented as a *point in a wellfounded relation*: consider the $\in$-relation on the transitive closure $\mathrm{TC}(\{x\})$ with distinguished element $x \in \mathrm{TC}(\{x\})$. By the MOSTOWSKI isomorphism theorem x is uniquely determined by the pair $(x,\mathrm{TC}(\{x\}))$ up to order isomorphism.

Definition 6.7.1: An ordered pair $x = (x,R_x)$ is a *point* if R_x is a wellfounded relation and $x \in \mathrm{dom}(R_x)$. Unless specified otherwise we use R_x to denote the wellfounded relation of the point x.

Obviously, $(x,\in\restriction \mathrm{TC}(\{x\}))$ is a point. Conversely, any point $x = (x,R_x)$ can be interpreted as a standard set $I(x)$. Define recursively

$$I_x : \mathrm{dom}(R_x) \rightarrow V,\ I_x(u) = \{I_x(v) | vRu\}.$$

Then let $I(x) = I_x(x)$ be the *interpretation* of x. Note that for points x and y

$$\begin{aligned} I_x(u) = I_y(v) \quad &\text{iff} \quad \{I_x(u') | u'R_xu\} = \{I_x(v') | v'R_yv\} \\ &\text{iff} \quad \forall u'R_xu\ \exists v'R_yv\ I_x(u') = I_y(v')) \wedge \\ &\qquad \wedge(\forall v'R_yv\ \exists u'R_xu\ I_x(u') = I_y(v'). \end{aligned}$$

This means that the relation $I_x(u) = I_y(v)$ in the variables u and v can be defined recursively without actually forming the interpretations $I_x(u)$ and $I_y(v)$. Wellfounded relations and points can be handled within the theory SO. This will allow to define a model of ZFC within SO. So assume SO for the following construction.

Definition 6.7.2: Define a relation $\equiv$ on points $x = (x, R_x), y = (y, R_y)$ by induction on the product wellorder $R_x \times R_y$:

$$(x, R_x) \equiv (y, R_y) \quad \text{iff} \quad \forall u R_x x\ \exists v R_y y\ (u, R_x) \equiv (v, R_y) \wedge \\ \wedge \forall v R_y y\ \exists u R_x x\ (u, R_x) \equiv (v, R_y).$$

Theorem 6.7.1: $\equiv$ is an equivalence relation on points.

Proof We only check transitivity; reflexivity and symmetry may be proved similarly.

Transitivity. Consider points $x = (x, R_x)$, $y = (y, R_y)$ and $z = (z, R_z)$. We show by induction on the wellfounded relation $R_x \times R_y \times R_z$ that

$$(u, R_x) \equiv (v, R_y) \wedge (v, R_y) \equiv (w, R_z) \rightarrow (u, R_x) \equiv (w, R_z).$$

Assume that the claim holds for all $u' R_x u$, $v' R_y v$ and $w' R_z w$. Assume that

$$(u, R_x) \equiv (v, R_y) \wedge (v, R_y) \equiv (w, R_z).$$

To show that $(u, R_x) \equiv (w, R_z)$ consider $u' R_x u$. By $(u, R_x) \equiv (v, R_y)$ take $v' R_y v$ such that $(u', R_x) \equiv (v', R_y)$. By $(v, R_y) \equiv (w, R_z)$ take $w' R_z w$ such that $(v', R_y) \equiv (w', R_z)$. By the inductive assumption, $(u', R_x) \equiv (v', R_y)$ and $(v', R_y) \equiv (w', R_z)$ imply that $(u', R_x) \equiv (w', R_z)$. Thus

$$\forall u' R_x u\ \exists w' R_z w\ (u', R_x) \equiv (w', R_z).$$

Similarly

$$\forall w' R_z w\ \exists u' R_x u\ (u', R_x) \equiv (w', R_z)$$

and thus $(u, R_x) \equiv (w, R_z)$. In particular for $x = (x, R_x)$, $y = (y, R_y)$ and $z = (z, R_z)$

$$x \equiv y \wedge y \equiv z \rightarrow x \equiv z.$$

□

We now define a membership relation for points.

Definition 6.7.3: Let $x = (x, R_x)$ and $y = (y, R_y)$ be points. Then set

$$x \blacktriangleleft y \text{ iff } \exists v R_y y\ x \equiv (v, R_y).$$

Lemma 6.7.1: The equivalence relation $\equiv$ is a congruence relation with respect to $\blacktriangleleft$, i.e.,

$$x \blacktriangleleft y \wedge x \equiv x' \wedge y \equiv y' \rightarrow x' \blacktriangleleft y'.$$

Proof Let $x \blacktriangleleft y \wedge x \equiv x' \wedge y \equiv y' \rightarrow x' \blacktriangleleft y'$. Take $v R_y y$ such that $x \equiv (v, R_y)$. By $y \equiv y'$ take $v' R_{y'} y'$ such that $v \equiv v'$. Since $\equiv$ is an equivalence relation, the relations $x \equiv x'$, $x \equiv v$ and $v \equiv v'$ imply $x' \equiv v'$. Hence $x' \blacktriangleleft y'$. □

6.8 The class of points satisfies ZFC

We show that the class $\mathbb{P}$ of points with the relations $\equiv$ and $\blacktriangleleft$ satisfies the axioms ZFC of Zermelo-Fraenkel set theory with the axiom of choice. For the existence axioms of ZFC we prove a lemma about combining points into a single point.

Lemma 6.8.1: (SO) Let $(x_i|i \in A)$ be a set-sized definable sequence of points, i.e., A is a set of ordinals and the function $i \mapsto x_i \in \mathbb{P}$ is definable. Then there is a point $y = (y, R_y)$ such that for all points x holds

$$x \blacktriangleleft y \text{ iff } \exists i \in A\ x \equiv x_i.$$

Proof For $i \in A$ let $x_i = (x_i, R_i)$. Define points $x'_i = (x'_i, R'_i)$ by "colouring" every element of $\mathrm{dom}(R_i)$ by the "colour" i:

$$x'_i = (i, x_i) \text{ and } R'_i = \{((i,\alpha),(i,\beta))|(\alpha,\beta) \in R_i\}.$$

The points (x'_i, R'_i) and (x_i, R_i) are isomorphic and so $(x'_i, R'_i) \equiv (x_i, R_i)$. We may thus assume that the domains of the wellfounded relations R_i are pairwise disjoint. Take some $y \notin \bigcup_{i\in A} \mathrm{dom}(R_i)$ and define the point $y = (y, R_y)$ by

$$R_y = \bigcup_{i\in A} R_i \cup \{(x_i, y)|i \in A\}.$$

Consider $i \in A$. If $x \in \mathrm{dom}(R_i)$ then the iterated R_i-predecessors of x are equal to the iterated R_y-predecessors of x. Hence $(x, R_i) \equiv (x, R_y)$.

Assume now that $x \blacktriangleleft y$. Take vR_yy such that $x \equiv (v, R_y)$. Take $i \in A$ such that $v = x_i$. By the previous remark

$$x \equiv (v, R_y) = (x_i, R_y) \equiv (x_i, R_i) = x_i.$$

Conversely consider $i \in A$ and $x \equiv x_i$. Then $x \equiv x_i = (x_i, R_i) \equiv (x_i, R_y)$ and x_iR_yy. This implies $x \blacktriangleleft y$. □

We are now able to canonically interpret the theory ZFC within SO.

Theorem 6.8.1: (SO) $\mathbb{P} = (\mathbb{P}, \equiv, \blacktriangleleft)$ is a model of ZFC.

Proof (1) The axiom of *extensionality* holds in $\mathbb{P}$:

$$\forall x \forall y(\forall z(z \blacktriangleleft x \leftrightarrow z \blacktriangleleft y) \rightarrow x \equiv y).$$

Proof. Consider points x and y such that $\forall z(z \blacktriangleleft x \leftrightarrow z \blacktriangleleft y)$. Consider uR_xx. Then $(u, R_x) \blacktriangleleft (x, R_x) = x$. By assumption, $(u, R_x) \blacktriangleleft (y, R_y)$. By definition take vR_yy such that $(u, R_x) \equiv (v, R_y)$. Thus

$$\forall uR_xx\ \exists vR_yy\ (u, R_x) \equiv (v, R_y).$$

By exchanging x and y one also gets

$$\forall vR_yy\ \exists uR_xx\ (u, R_x) \equiv (v, R_y).$$

Hence $x \equiv y$. *qed*(1)

(2) The axiom of *pairing* holds in $\mathbb{P}$:

$$\forall x \forall y \exists z \forall w (w \blacktriangleleft z \leftrightarrow (w \equiv x \vee w \equiv y)).$$

Proof. Consider points $x = (x, R_x)$ and $y = (y, R_y)$. By the comprehension lemma 6.8.1 there is a point $z = (z, R_z)$ such that for all points w

$$w \blacktriangleleft z \leftrightarrow (w \equiv x \vee w \equiv y).$$

qed(2)

(3) The axiom of *unions* holds in $\mathbb{P}$:

$$\forall x \exists y \forall z (z \blacktriangleleft y \leftrightarrow \exists w (w \blacktriangleleft x \wedge z \blacktriangleleft w)).$$

Proof. Consider a point $x = (x, R_x)$. Let

$$A = \{i \in \mathrm{dom}(R_x) | \exists u \in \mathrm{dom}(R_x)\ iR_xuR_xx\}.$$

For $i \in A$ define the point $x_i = (i, R_x)$. By the Comprehension Lemma 6.8.1 there is a point $y = (y, R_y)$ such that for all points z

$$z \blacktriangleleft y \leftrightarrow \exists i \in A\ z \equiv x_i.$$

To show the axiom consider some $z \blacktriangleleft y$. Take $i \in A$ such that $z \equiv x_i$. Take $u \in \mathrm{dom}(R_x)$ such that iR_xuR_xx. Then $z \equiv x_i = (i, R_x) \blacktriangleleft (u, R_x) \blacktriangleleft (x, R_x) = x$, i.e., $\exists w (z \blacktriangleleft w \blacktriangleleft x)$.

Conversely assume that $\exists w (z \blacktriangleleft w \blacktriangleleft x)$ and take w such that $z \blacktriangleleft w \blacktriangleleft x$. Take uR_xx such that $w \equiv (u, R_x)$. Then $z \blacktriangleleft (u, R_x)$. Take iR_xu such that $z \equiv (i, R_x) = x_i$. Then $z \blacktriangleleft y$. *qed*(3)

(4) The *replacement schema* holds in $\mathbb{P}$, i.e., for every first-order formula $\varphi(u, v)$ in the language of $\equiv$ and $\blacktriangleleft$ the following is true in $\mathbb{P}$:

$$\forall u, v, v' ((\varphi(u, v) \wedge \varphi(u, v')) \rightarrow v \equiv v') \rightarrow \forall x \exists y \forall z (z \blacktriangleleft y \leftrightarrow \exists u (u \blacktriangleleft x \wedge \varphi(u, z))).$$

Proof. Note that the formula φ may contain further free parameters, which we do not mention for the sake of simplicity. Assume that $\forall u, v, v' ((\varphi(u, v) \wedge \varphi(u, v')) \rightarrow v \equiv v')$ and let $x = (x, R_x)$ be a point. Let $A = \{i | iR_xx\}$. For each $i \in A$ we have the point $(i, R_x) \blacktriangleleft (x, R_x) = x$. Using replacement and choice in SO we can pick for each $i \in A$ a point $z_i = (z_i, R_{z_i})$ such that $\varphi((i, R_x), z_i)$ holds if such a point exists. By the Comprehension Lemma 6.8.1 there is a point $y = (y, R_y)$ such that for all points z

$$z \blacktriangleleft y \leftrightarrow \exists i \in A\ z \equiv z_i.$$

To show the instance of the replacement schema under consideration, assume that $z \blacktriangleleft y$. Take $i \in A$ such that $z \equiv z_i$. Then $(i, R_x) \blacktriangleleft (x, R_x) = x$, $\varphi((i, R_x), z_i)$ and $\varphi((i, R_x), z)$. Hence $\exists u (u \blacktriangleleft x \wedge \varphi(u, z))$.

Conversely, assume that $\exists u (u \blacktriangleleft x \wedge \varphi(u, z))$. Take $u \blacktriangleleft x$ such that $\varphi(u, z)$. Take iR_xx, $i \in A$ such that $u \equiv (i, R_x)$. Then $\varphi((i, R_x), z)$. By definition of z_i, $\varphi((i, R_x), z_i)$. The functionality of the formula φ implies $z \equiv z_i$. Hence $\exists i \in A\ z \equiv z_i$ and $z \blacktriangleleft y$. *qed*(4)

The replacement schema also implies the *separation* schema.

(5) The axiom of *powersets* holds in $\mathbb{P}$:

$$\forall x \exists y \forall z (z \blacktriangleleft y \leftrightarrow \forall w (w \blacktriangleleft z \rightarrow w \blacktriangleleft x)).$$

Proof. By the separation schema it suffices to show that

$$\forall x \exists y \forall c (\forall w (w \blacktriangleleft c \rightarrow w \blacktriangleleft x) \rightarrow c \blacktriangleleft y).$$

Consider a point $x = (x, R_x)$. Let $F = \mathrm{dom}(R_x) \cup \mathrm{ran}(R_x)$ be the *field* of R_x. By the powerset axiom of SO choose some set P such that $\mathrm{Pow}(P, F)$:

$$\forall z (\exists \alpha (\alpha \in z) \wedge \forall \alpha (\alpha \in z \rightarrow \alpha \in F) \rightarrow \exists \xi \forall \beta (\beta \in z \leftrightarrow (\beta, \xi) \in P)).$$

Choose two large ordinals δ and y such that

$$\forall \alpha \in F \ \alpha < \delta \text{ and } \forall \xi (\xi \in \mathrm{ran}(P) \rightarrow (\delta, \xi) < y).$$

Define a point $y = (y, R_y)$ by

$$R_y = R_x \cup \{(\beta, (\delta, \xi)) | (\beta, \xi) \in P\} \cup \{((\delta, \xi), y) | \xi \in \mathrm{ran}(P)\}.$$

To show the axiom consider some point $c = (c, R_c)$ such that $\forall w (w \blacktriangleleft c \rightarrow w \blacktriangleleft x)$. Define a corresponding subset z of F by

$$z = \{\beta \in F | \exists v R_c c \ (v, R_c) \equiv (\beta, R_x)\}.$$

We may assume for simplicity that $z \neq \emptyset$. By the powerset axiom of SO choose $\xi \in \mathrm{ran}(P)$ such that

$$\forall \beta (\beta \in z \leftrightarrow (\beta, \xi) \in P).$$

We claim that $((\delta, \xi), R_y) \equiv c$ and thus $c \blacktriangleleft y$.

Consider $\beta R_y (\delta, \xi)$. By the definition of $R_y$we have $(\beta, \xi) \in P$ and so $\beta \in z$. By the definition of z choose $v R_c c$ such that $(v, R_c) \equiv (\beta, R_x) \equiv (\beta, R_y)$.

Conversely, consider $v R_c c$. Then $(v, R_c) \blacktriangleleft (c, R_c) = c$. The subset property implies $(v, R_c) \blacktriangleleft (x, R_x) = x$. Take $\beta R_x x$ such that $(v, R_c) \equiv (\beta, R_x) \equiv (\beta, R_y)$. By definition, $\beta \in z$, $(\beta, \xi) \in P$ and $\beta R_y (\delta, x)$. *qed*(5)

(6) The *axiom of choice* holds in $\mathbb{P}$:

$$\forall x ((\forall y, z (y \blacktriangleleft x \wedge z \blacktriangleleft x \rightarrow (\exists u \ u \blacktriangleleft y \wedge (\neg y \equiv z \rightarrow \neg \exists u (u \blacktriangleleft y \wedge u \blacktriangleleft z))))) \quad \rightarrow$$
$$\rightarrow \exists w \forall y (y \blacktriangleleft x \rightarrow \exists u ((u \blacktriangleleft w \wedge u \blacktriangleleft y) \wedge \forall v ((v \blacktriangleleft w \wedge v \blacktriangleleft y) \rightarrow u \equiv v)))).$$

Proof. Let $x = (x, R_x) \in \mathbb{P}$ be a point such that

$$\forall y, z ((y \blacktriangleleft x \wedge z \blacktriangleleft x) \rightarrow (\exists u \ u \blacktriangleleft y \wedge (\neg y \equiv z \rightarrow \neg \exists u (u \blacktriangleleft y \wedge u \blacktriangleleft z)))).$$

Choose an ordinal $\alpha \in \mathrm{dom}(R_x)$ and define the point $w = (\alpha, R_w)$ by letting its "elements" be least ordinals in the "elements" of x:

$$\begin{aligned} R_w \quad = \quad & R_x \cup \{(\xi, \alpha) | \exists \zeta (\xi R_x \zeta R_x x \wedge \\ & \wedge (\forall \xi' < \xi \ \forall \zeta' ((\zeta, R_x) \equiv (\zeta', R_x) \rightarrow \neg (\xi R_x \xi' R_x \zeta))))\}. \end{aligned}$$

To show that w witnesses the axiom of choice for x consider a point y with $y \blacktriangleleft x$. We may assume that y is of the form $y = (\zeta, R_x)$ where $\zeta R_x x$. By the assumption on x there exists $u \blacktriangleleft y$. Take some ξ such that $(\xi, R_x) \equiv u$. We

may assume that ζ and ξ with these properties are chosen so that ξ is minimal in the ordinals. Then

$$\xi R_x \zeta R_x x \wedge (\forall \xi' < \xi \; \forall \zeta'((\zeta, R_x) \equiv (\zeta', R_x) \rightarrow \neg(\xi R_x \xi' R_x \zeta))) \tag{6.1}$$

and so $\xi R_w \alpha$. Thus $u \blacktriangleleft w$. To show the uniqueness of this u with $u \blacktriangleleft w \wedge u \blacktriangleleft y$ consider some v with $v \blacktriangleleft w \wedge v \blacktriangleleft y$. We may assume that v is of the form $v = (\xi', R_w)$ with $\xi' R_w \alpha$. By the definition of R_w we choose some ζ' such that

$$\xi' R_x \zeta' R_x x \wedge (\forall \xi'' < \xi' \; \forall \zeta''((\zeta', R_x) \equiv (\zeta'', R_x) \rightarrow \neg(\xi' R_x \xi'' R_x \zeta'))). \tag{6.2}$$

Now

$$v \blacktriangleleft y \blacktriangleleft x \text{ and } v = (\xi', R_w) \blacktriangleleft (\zeta', R_w) \blacktriangleleft (x, R_x) = x.$$

Since the "elements" of x are "pairwise disjoint", we have $y \equiv (\zeta', R_w)$. Since $y \equiv (\zeta, R_x)$ the conditions (2) and (3) become equivalent and define the same ordinal $\xi = \xi'$. Hence

$$u \equiv (\xi, R_x) \equiv (\xi', R_w) \equiv v.$$

qed(6)

(7) The *foundation schema* holds in $\mathbb{P}$, i.e., for every first-order formula $\varphi(u)$ in the language of $\equiv$ and $\blacktriangleleft$ the following is true in $\mathbb{P}$:

$$\exists u \; \varphi(u) \rightarrow \exists y(\varphi(y) \wedge \forall z(z \blacktriangleleft y \rightarrow \neg\varphi(z)).$$

Proof. Note that the formula φ may contain further free parameters, which we do not mention for the sake of simplicity. Assume that $\exists u \; \varphi(u)$. Take a point $x = (x, R_x)$ such that $\varphi(x)$. Since R_x is wellfounded one may take an R_x-minimal $y \in \mathrm{dom}(R_x)$ such that $\varphi((y, R_x))$. Letting y also denote the point (y, R_x) then $\varphi(y)$. To prove the axiom, consider some point $z \blacktriangleleft y$. Take $v R_x y$ such that $z \equiv (v, R_x)$. By the R_x-minimal choice of y we have $\neg\varphi((v, R_x))$. Hence $\neg\varphi(z)$. *qed*(7)

(8) The axiom of *infinity* holds in $\mathbb{P}$, i.e.,

$$\exists x((\exists y \; y \blacktriangleleft x) \wedge (\forall y(y \blacktriangleleft x \rightarrow \exists z(z \blacktriangleleft x \wedge \forall u(u \blacktriangleleft z \leftrightarrow (u \blacktriangleleft y \vee u \equiv y))))))$$

Proof. In SO let ω be the smallest limit ordinal. We show that

$$x = (\omega, <\restriction (\omega + 1)^2)$$

witnesses the axiom. Since $(0, <\restriction (\omega+1)^2) \blacktriangleleft (\omega, <\restriction (\omega+1)^2)$ we have $\exists y \; y \blacktriangleleft x$. Consider some $y \blacktriangleleft x$. We may assume that $y = (n, <\restriction (\omega+1)^2)$ for some $n < \omega$. Set

$$z = (n + 1, <\restriction (\omega + 1)^2).$$

It is easy to check that

$$z \blacktriangleleft x \wedge \forall u(u \blacktriangleleft z \leftrightarrow (u \blacktriangleleft y \vee u \equiv y)).$$

qed(8) □

6.9 T codes a model of SO

The truth predicate T contains information about a large class of sets of ordinals.

Definition 6.9.1: For ordinals μ and α define

$$X(\mu, \alpha) = \{\beta < \mu | T(G(\alpha, \beta))\}.$$

Set

$$\mathcal{S} = \{T(\mu, \alpha) | \mu, \alpha \in \mathrm{Ord}\}.$$

Theorem 6.9.1: $(\mathrm{Ord}, \mathcal{S}, <, =, \in, G)$ is a model of the theory SO.

Proof The axioms (1)-(7) are obvious. The proofs of axiom schemas (8) and (9) rest on a LEVY-type reflection principle. For $\theta \in \mathrm{Ord}$ define

$$\mathcal{S}_\theta = \{X(\mu, \alpha) | \mu, \alpha \in \theta\}.$$

Then for any L_{SO}-formula $\varphi(v_0, \ldots, v_{n-1})$ and $\eta \in \mathrm{Ord}$ there is some limit ordinal $\theta > \eta$ such that

$$\begin{aligned}\forall \xi_0, \ldots, \xi_{n-1} \in \theta((\mathrm{Ord}, \mathcal{S}, <, =, \in, G) \vDash \varphi[\xi_0, \ldots, \xi_{n-1}] \text{ iff} \\ \text{iff } (\theta, \mathcal{S}_\theta, <, =, \in, G) \vDash \varphi[\xi_0, \ldots, \xi_{n-1}]).\end{aligned}$$

Since all elements of $\mathcal{S}_\theta$ can be defined from the truth function T and ordinals $< \theta$, the right-hand side can be evaluated in the structure $(\theta, <, G \cap \theta^3, T)$ by an L_R-formula φ^* which can be recursively computed from φ. Hence

$$\begin{aligned}\forall \xi_0, \ldots, \xi_{n-1} \in \theta((\mathrm{Ord}, \mathcal{S}, <, =, \in, G) \vDash \varphi[\xi_0, \ldots, \xi_{n-1}] \text{ iff} \\ \text{iff } (\theta, <, G \cap \theta^3, T) \vDash \varphi^*[\xi_0, \ldots, \xi_{n-1}]).\end{aligned}$$

So sets witnessing axioms (8) and (9) can be defined over $(\theta, <, G \cap \theta^3, T)$ and are thus elements of $\mathcal{S}$.

The powerset axiom of SO can be shown by a similar reflection argument. □

6.10 Ordinal computability corresponds to constructibility

KURT GÖDEL [4] defined the inner model L of *constructible sets* as the union of a hierarchy of levels L_α:

$$L = \bigcup_{\alpha \in \mathrm{Ord}} L_\alpha$$

where the hierarchy is defined by: $L_0 = \emptyset$, $L_\delta = \bigcup_{\alpha < \delta} L_\alpha$ for limit ordinals δ, and $L_{\alpha+1}$ = the set of all sets which are first-order definable in the structure $(L_\alpha, \in)$. The model L is the $\subseteq$-smallest inner model of set theory. The standard reference for the theory of the model L is the monograph [3].

The following main result provides a characterization of ordinal register computability which does not depend on a specific machine model or coding of language:

Theorem 6.10.1: A set x of ordinals is ordinal computable if and only if it is an element of the constructible universe L.

Proof Let $x \subseteq \mathrm{Ord}$ be ordinal computable by the program P from the ordinals $\delta_1, \ldots, \delta_{n-1}$, so that for every $\alpha \in \mathrm{Ord}$:

$$P : (\alpha, \delta_1, \ldots, \delta_{n-1}, 0, 0, \ldots) \mapsto \chi_x(\alpha).$$

By the simple nature of the computation procedure the same computation can be carried out inside the inner model L, so that for every $\alpha \in \mathrm{Ord}$:

$$(L, \in) \vDash P : (\alpha, \delta_1, \ldots, \delta_{n-1}, 0, 0, \ldots) \mapsto \chi_x(\alpha).$$

Hence $\chi_x \in L$ and $x \in L$.

Conversely consider $x \in L$. Since $(\mathrm{Ord}, \mathcal{S}, <, =, \in, G)$ is a model of the theory SO there is an inner model M of set theory such that

$$\mathcal{S} = \{z \subseteq \mathrm{Ord} \,|\, z \in M\}.$$

Since L is the $\subseteq$-smallest inner model, $L \subseteq M$. Hence $x \in M$ and $x \in \mathcal{S}$. Let $x = X(\mu, \alpha)$. By the computability of the truth predicate, x is ordinal register computable from the parameters μ and α. □

References

[1] the author. Ordinal computers. arXiv:math.LO/9804076, 1998.

[2] Nigel J. Cutland. *Computability: An introduction to Recursive Function Theory*. Perspectives in Mathematical Logic. Cambridge University Press, 1980.

[3] Keith Devlin. *Constructibility*. Perspectives in Mathematical Logic. Springer-Verlag, Berlin, 1984.

[4] Kurt Gödel. *The Consistency of the Continuum Hypothesis*, volume 3 of *Ann. of Math. Studies*. Princeton University Press, Princeton, 1940.

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

[6] Thomas Jech. *Set Theory. The Third Millennium Edition*. Springer Monographs in Mathematics. Springer-Verlag, 2003.

[7] Peter Koepke. Turing computations on ordinals. *Bull. Symbolic Logic*, 11(3):377–397, 2005.

7. EHRENFEUCHT–FRAïSSÉ GAMES ON LINEAR ORDERS

abstract

We write strategies for both players in the Ehrenfeucht-Fraisse game played between two linear orders. We prove that our strategies win whenever possible. The strategy in a game of fixed, finite length focuses the attention of both players on certain finite sets called "total information." These finite sets are semimodels in the sense that if one player does not have a winning strategy, then the other player can win, even when forced to choose only elements of a model which are indicated by the total information.

These sets are semimodels– finite, nested models, such that a formula is satisfied in a linear order iff the formula is semimodel-satisfied in the corresponding semimodel. The strategy implies the decidability of the theory of linear order, and gives a completion of any formula by one with quantifier rank only 2 larger than the original formula. The strategy generalizes directly to the infinitary theory of linear order.

7.1 "Total information" is sufficient for the EF game

For any linear order λ, let λ^+ be the ordered set of cuts in λ, i.e.

Definition 7.1.1: $\lambda^+ = (\{(X,Y) : X \cup Y = \lambda, \forall x \in X \forall y \in Y x < y\}; (X,Y) \leq (W,Z)$ iff $X \subseteq W)$.

Definition 7.1.2: For any linear order λ and element x, let $Th_k^{\mathrm{loc}}(\lambda, x)$ be {formulas ϕ of quantifier rank k with the single variable v free : $\exists E\ \forall A, B, C, D (A + E + B + \lambda + C + E + D \models_{[x/v]} \phi(v))$} where A, B, C, D, E range over linear orders.

Our goal is to prove theorem 7.1.1, i.e., to find in any linear order λ and for any natural number k a finite set $I_k(\lambda) \subseteq \lambda^+$ such that $\lambda \equiv_k \lambda'$ iff $I_k(\lambda) = I_k(\lambda')$ and the same $k-1$-quantifier local types $Th_{k-1}^{\mathrm{loc}}(\lambda, x)$ are realized in both λ and λ' at and between the elements of I_k.

Definition 7.1.3: Let Seq be the set of decreasing sequences of natural numbers. For k any natural number, let Seq(k) be the set of descending sequences of natural numbers $\leq k-2$. Let $<^{\mathrm{seq}}$ be lexicographical ordering on Seq.

Definition 7.1.4: For λ any linear order and $(a,b) = ((L_0, L_1), (R_0, R_1))$ any pair of adjacent elements of λ^+, define $\lambda^{(a,b)} = \{x \in \lambda : (x \in L_1 \wedge \exists z (z \in L_1 \wedge z < x)) \wedge (x \in R_0 \wedge \exists z (z \in R_0 \wedge z > x))\}$.

Definition 7.1.5: For λ any linear order, we define $I_\emptyset(\lambda) = \emptyset$. If $s \in$ Seq and $I_s(\lambda)$ is defined and $s <^{\mathrm{seq}} s'$ and s' is the $<^{\mathrm{seq}}$-successor of s, let l be the least

member of s', and let $I_{s'}(\lambda)$ contain $I_s(\lambda)$ and for each adjacent pair of elements of $I_s(\lambda)$, elements of λ^+ representing the appearance and disappearance of each local l-quantifier type. Precisely:

For each pair $(a,b), a < b$, of adjacent elements of $I_s(\lambda)$ and for each $t \in \{Th_l^{\mathrm{loc}}(\lambda, z) : z \in \lambda^{(a,b)}\}$, define the cuts $(A_t^{(a,b)}, B_t^{(a,b)}) \in \lambda^+$ and $(C_t^{(a,b)}, D_t^{(a,b)}) \in \lambda^+$ as follows:

- $A_t^{(a,b)} = \{x \in \lambda : \forall z \in \lambda^{(a,b)} t(z) \to z > x)\}$,
- $B_t^{(a,b)}$ = the complement of A_t, i.e. $\{x \in \lambda : \exists z \in \lambda^{(a,b)}(t(z) \wedge z \leq x)\}$,
- $C_t^{(a,b)} = \{x \in \lambda : \exists z \in \lambda^{(a,b)}(t(z) \wedge z \geq x)\}$,
- $D_t^{(a,b)}$ = the complement of C_t, i.e. $\{x \in \lambda : \forall z \in \lambda^{(a,b)}(t(z) \to z < x)\}$.

If the cut $a = (L_0, L_1) \in \lambda^+$ was created as $(C_u^{(a_1,b_1)}, D_u^{(a_1,b_1)})$ (rather than $(A_u^{(a_1,b_1)}, B_u^{(a_1,b_1))})$) and $u = Th_m^{\mathrm{loc}}(\lambda, w)$ and $m > l$ and $\forall x \in C_u \exists y \in C_u(y > x \wedge t(y))$, then we say a covers t. Similarly, if $b = (R_0, R_1)$ was created as $(A_u^{(a_1,b_1)}, B_u^{(a_1,b_1))})$ (rather than $(C_u^{(a_1,b_1)}, D_u^{(a_1,b_1)})$) and u is a local type of higher quantifier rank than t and $\forall x \in B_u \exists y \in B_u(y < x \wedge t(y))$, then we say b covers t.

Let $I_{s',(a,b)} = \{(A_t^{(a,b)}, B_t^{(a,b)}) : (\exists z \in \lambda^{(a,b)}(t = Th_l^{\mathrm{loc}}(\lambda, z))) \wedge a$ does not cover $t\} \cup \{(C_t^{(a,b)}, D_t^{(a,b)}) : (\exists z \in \lambda^{(a,b)}(t = Th_l^{\mathrm{loc}}(\lambda, z))) \wedge b$ does not cover $t\}$ be the set of first and last appearances of each local type t in the interval $\lambda^{(a,b)}$.

Let $I_{s'} = I_s \cup \cup\{I_{s',(a,b)} : (a,b)$ is a pair of adjacent elements of $I_s\}$.

For any $k \geq 2$, let $I_k(\lambda) = \cup I_s(\lambda) : \lambda \in Seq(k)$. (This choice of index is made because of lemma 7.1.1.)

Definition 7.1.6: For any linear orders λ and λ' and s a decreasing sequence of natural numbers, we define $\lambda \equiv_s \lambda'$ by induction on s. Say $\lambda \equiv_\emptyset \lambda'$ holds for any linear orders. Let $s \in$ Seq and s' be the $<^{\mathrm{seq}}$ successor of $s$and let l be the least number in s' as in definition 7.1.4. Then $\lambda \equiv_{s'} \lambda'$ iff $\lambda \equiv_s \lambda'$ and for any pair of adjacent elements of I_s $(a,b) = ((L_0, L_1), (R_0, R_1))$, the cuts of definition 7.1.5, for any two types $t, u \in \{Th_l^{\mathrm{loc}}(\lambda, z) : z \in \lambda^{(a,b)}\}$ satisfy each of the following conditions in λ iff they satisfy the same condition in λ':

1. $(A_t^{(a,b)}, B_t^{(a,b)}) = (L_0, L_1) = a$ or $(C_t^{(a,b)}, D_t^{(a,b)}) = (R_0, R_1) = b$.
2. $B_t^{(a,b)} \cap C_t^{(a,b)}$ is a singleton and $\neg(a$ covers t and b covers $t)$.
3. $B_t^{(a,b)}$ has a least element, and a does not cover t.
4. $C_t^{(a,b)}$ has a greatest element, and b does not cover t.
5. $\exists z \in (B_t^{(a,b)} \setminus B_u^{(a,b)})$ and a does not cover u.
6. $\exists z \in (B_t^{(a,b)} \cap C_u^{(a,b)})$ and $\neg(a$ covers t and b covers $u)$.
7. $\exists z \in A_t^{(a,b)} \cap D_u^{(a,b)}$ and a doesn't cover t and b doesn't cover u.
8. $B_t^{(a,b)} \cap C_t^{(a,b)} = \emptyset$.

Lemma 7.1.1: If $\lambda \equiv_k \lambda'$, then for all $s \in \mathrm{Seq}(k)$, $\lambda \equiv_s \lambda'$.

Proof: Suppose $\lambda \equiv_s \lambda'$, but $\lambda \not\equiv_{s'} \lambda'$, where s' is the $<^{\mathrm{seq}}$-sucessor of s. We show that player I has a winning strategy in $EF_k(\lambda, \lambda')$, hence $\lambda \not\equiv_k \lambda'$.

We first prove, by induction, claim k: If, in the game $EF_k(\lambda, \lambda')$, player I plays the first move in (a_s, b_s) for all $s \in \mathrm{Seq}(k)$, where if s' is the $<^{\mathrm{seq}}$-sucessor of s and $s < s' \in \mathrm{Seq}(k)$, then $\{a_{s'}, b_{s'}\} \subseteq I_{s'} \setminus I_s$ and $a_{s'} = (L(a_{s'}), R(a_{s'}))$ and is defined in $I_{s'}$ as $(A^{(a_s,b_s)}_{u(a_{s'})}, B^{(a_s,b_s)}_{u(a_{s'})})$ where $u(a_{s'})$ was not covered by a_s, or $a_{s'} = (C^{(a_s,b_s)}_{u(a_{s'})}, D^{(a_s,b_s)}_{u(a_{s'})})$ where $u(b_{s'})$ was not covered by b_s, then player II must answer in (a_s, b_s), for all $s \in \mathrm{Seq}(k)$.

Proof: suppose player II answer inside (a_r, b_r), for all $r \leq s$, but outside $(a_{s'}, b_{s'})$. Without loss of generality, suppose player I plays x_0 above $a_{s'}$, and player II plays y_0 below $a_{s'}$. Writing $a_s = (L(a_{s'}), R(a_{s'}))$, this means that player I played in $R(a_{s'})$, and player II played in $L(a_{s'})$.

Let the least number in s' be l, so that $u(a_{s'}) = Th^{\mathrm{loc}}_l(\lambda, x)$ is a local type of quantifier depth l. If x_0 is the least element of $R(a_{s'})$, or the greatest element of $L(a_{s'})$, then player II has lost, for then x_0 has quantifier-rank-l local type $u(a_{s'})$, and there is no $y \in \lambda'^{(a_s,b_s)}$ such that $Th^{\mathrm{loc}}_k(\lambda', y) = u(a_{s'})$ and $y < a_{s'}$, by the formula defining A and B or C and D in definition 7.1.5.

Let the greatest number in s' be $j \leq k-2$; let j^- be the $<^{\mathrm{seq}}$-precessor of (j), with domain $\{i : i < j\}$. Player I plays x_1 at an element of λ of type $u(a_{(j)})$ near $a_{(j)}$ in (a_{j^-}, b_{j^-}), where "near" means: if $a_j = (A^{(a_{j^-},b_{j^-})}_{u(a_{(j)})}, B^{(a_{j^-},b_{j^-})}_{u(a_{(j)})})$, then player I plays $x_1 \geq a_j$ in λ and x_1 should be the least element of $B^{(a_{j^-},b_{j^-})}_{u(a_j)}$ if there is one, and otherwise, for each $t \in \{Th^{\mathrm{loc}}_{j-1}(\lambda, z) : z \in \lambda\}$ (a finite set) which is realized for some $z \in \lambda^{(a_{j^-},b_{j^-})}$, if realizations of t are bounded from below in $B^{(a_{j^-},b_{j^-})}_{u(a_j)}$, then all of those realizations occur above x_1, and if t is realized below every $y \in B^{(a_{j^-},b_{j^-})}_{u(a_j)}$, then t is realized a large number of times (say, $10 + 2^{10+2^{10+k}}$ or more) above x_1. This is done to assure that $I_{(j)\cup r}(\lambda^{>x_1})$ will contain an $\equiv_{(j)}$ copy of $I_{(j)\cup r}(\lambda^{>a_s})$ whenever r is a descending sequence of numbers $< j$. On the other hand, if $a_j = (C^{(a_{j^-},b_{j^-})}_{u(a_{(j)})}, D^{(a_{j^-},b_{j^-})}_{u(a_{(j)})})$, then player I plays $x_1 \leq a_j$ in λ' near the cut $a_{(j)}$, where, again, "near" means that x_1 bounds to the left all quantifier-rank $< j$ local types which can be bound to the left.

If player I plays $x_1 < a_{(j)} = (C, D) < y_0 \in \lambda'$, and player II plays y_1 in the interval (a_{j^-}, b_{j^-}) and $y_1 < a_{(j)} \in I_{(j)}$, then we may consider the game $EF_{k-1}(\lambda^{>y_1}, \lambda'^{>x_1})$, in which player I has played x_0, and player II has answered with y_0. We will show that this violates the induction hypothesis. We check, for r any decreasing sequence of numbers $< j$, that the set I_r defined in $\lambda^{>x_1}$ and $\lambda'^{>y_1}$ are $\equiv_{(j)}$ to the sets $I_(k \cup r)$ defined in λ and λ' to the right of $a_{(j)}$. In this case, x_1 was played close to a_s so that the only $t \in \{Th^{\mathrm{loc}}_{j-1}(\lambda, z) : z \in \lambda\}$ which are realized to the right of x_1 are those t which are realized to the right of a_s or which are covered by a_s. A cut c enters $I_(k \cup r)$ only if it was not covered by $a_{(j)}$. Therefore, player I was able to play $x_1 < a_{(j)}$ large enough that no elements of type c exist in $(x_1, a_{(j)})$, when $c \in I_(k \cup r)$. So, by claim $k-1$, player II loses. If player I plays $x_1 < a_{(s)} = (C, D) < y_0 \in \lambda'$, and player II plays y_1 outside the interval (a_{j^-}, b_{j^-}), then $y_1 < a_{j^-}$, then we may consider

the game $EF_{k-1}(\lambda^{<a_{(j)}}, \lambda'^{<a_{(j)}})$ in which player I has played x_1 and player II has played y_1. Player II's move disrespects I_s, for $s \in \mathrm{Seq}(j)$. If player I plays $x_1 > a_{(j)} = (A, B) \in \lambda$, of the type for which $a_{(j)}$ was defined, and player II plays $y_1 < a_{(j)} \in \lambda$, then we know that y_1 is not of the same j-quantifier local type as x_1. This concludes the proof of claim k.

Now we assume $\lambda \equiv_s \lambda'$ and that player II must answer player I's moves at the same elements or in the same interval between elements of I_s, and we show that $\lambda \not\equiv_{s'} \lambda'$ gives player I a winning strategy. The definition of $\equiv_{s'}$ lists conditions under which it fails, so we show that player I can exploit each of those eventualities and win. Suppose that $\equiv_{s'}$ fails in the interval (a, b) of I_s.

Case 1: Suppose $(A_t^{(a,b)}, B_t^{(a,b)}) = (L_0, L_1)$ in λ, but not in λ'. That is, elements of type t exist below any $z \in \lambda^{(a,b)}$. Then player I plays $x_0 \in \lambda'^{(a,b)}$ below every element of type t in $\lambda'^{(a,b)}$, where t has quantifier rank l, the least element of s'. By Claim k, player II answers with $y_0 \in \lambda'^{(a,b)}$. Now, for each initial segment s_j of s the least element of which is $> l$, player I plays x_j near a_{s_j} in $(a_{(s_j)^-}, b_{(s_j)^-})$, as described in claim k, and player II answers, by claims $k-1 \ldots k-j$, in the same intervals. After both players have played x_j and y_j close to a_{s_j}, for all initial segments s_j, there is an element z of type t below y_0 and above x_j or y_j (say, x_j) in λ'. There may well be an element z' of type t below x_0 and above y_j in λ, but for any such z', $I_l(y_j, z') \not\equiv_l I_l(y_j, a_s)$. So player I plays $x_{j+1} = z$, and as the remaining l-many moves would prove that $Th_l(\lambda, y_{j+1}) \neq t$, player II plays some z' of type t below a_{s_j}. Player I proceeds, as in claim l, to prove that $I_l(y_j, z') \not\equiv_l I_l(y_j, a_s)$.

As a result of case 1, we could add the condition – a covers t – to the list of the conditions in definition 7.1.6, the definition of $\lambda \equiv_s \lambda'$.

Case 2: Suppose z is the only element of $B_t^{(a,b)} \cap C_t^{(a,b)}$ in λ and that there are two elements $z_0 < z_1$ of $B_t^{(a,b)} \cap C_t^{(a,b)}$ in λ'. If a does not cover t, player I plays $x_0 = z_1$. If b does not cover t, player I plays $x_0 = z_0$. If t has quantifier-rank l, then the next $k-l-2$-many moves are spent isolating (a, b), after which player I plays $x_{k-l-1} = z_0$ after which player I spends the remaining l-many moves defining $I_l(z_s, z_0)$, where $z_s = x_j$ or $z_s = y_j$, z_s is near the cut $(L_0, L_1) = a_s$, and since a_s did not cover t, the interval (z_s, z) has no realization of t.

Case 3: Suppose $B_t^{(a,b)}$ has a least element, $z \in \lambda^{(a,b)}$, but no least element in $\lambda'^{(a,b)}$ and a does not cover t. Player I plays $x_0 = z$, and then proves that (a, y_0) contains an element of type t, as in the previous case.

Case 5: Suppose $z \in (B_t^{(a,b)} \setminus B_u^{(a,b)})$ in λ but $B_t^{(a,b)} = B_u^{(a,b)}$ in λ', and a does not cover u. Player I plays $x_0 = z$, and then proves that (a, y_0) contains an element of type u, as in case 2.

Case 6: If b doesn't cover u, player I plays $x_0 =$ an element of type t, $x_0 \in (B_t^{(a,b)} \cap C_u^{(a,b)})$ and shows $\exists z \in (x_0, b)$ of type u. If a doesn't cover t, player I plays $x_0 =$ an element of type u, $x_0 \in (B_t^{(a,b)} \cap C_u^{(a,b)})$ and shows $\exists z \in (a, x_0)$ of type t.

Case 7: Player I plays $x_0 = z$ and shows (x_0, b) contains no element of type u and (a, x_0) contains no element of type t.

Case 8: This holds iff $\neg\exists z(z \in \lambda^{(a,b)} \wedge t(z))$. By claim k, player I can play the element of type t in the model in which such an element exists, and then show that player II has replied outside $\lambda^{(a,b)}$. □

Definition 7.1.7: For λ any linear order and k any number, let t be the function with domain $\{\min(B_t) : (A_t, B_t) \in I_k \text{ and } B_t \text{ has a minimal element}\} \cup \{\max(C_t) : (C_t, D_t) \in I_k \text{ and } C_t \text{ has a maximal element}\}$, taking the value $Th^{\text{loc}}_{k-1}(\lambda, x)$ on argument x; let g be the function with domain I_k^+ taking the value $\{Th^{\text{loc}}_{k-1}(\lambda, x) : x \in \lambda^{(a,b)}\}$ on argument (a, b). We call I_k, t, g total information for λ, k.

Lemma 7.1.2: If $x \in \lambda$, then from $Th^{\text{loc}}_{k-1}(\lambda, x)$ and total information for λ, k we can determine total information for $\{y \in \lambda : y < x\}, k-1$.

Proof: Every element of $I_{k-1}(\{y \in \lambda : y < x\})$ is an element of $I_k(\lambda)$ or an element of $\cap_{A,B} I_{k-1}(A + E + B + \{y \in \lambda : y < x\})$. The value of t on $z \in I_{k-1}(\{y \in \lambda : y < x\}) \cap I_k(\lambda)$ is determined by finding $Th^{\text{loc}}_{k-2}(\lambda, z)$ from $Th^{\text{loc}}_{k-1}(\lambda, z)$; the value of t on $z \in I_{k-1}(\{y \in \lambda : y < x\}) \cap \cap_{A,B} I_{k-1}(A + E + B + \{y \in \lambda : y < x\})$ is the value which the intersection of total information for any large linear order which ends in a copy of $\{y \in \lambda, : y < x\}$, i.e, $\cap_{A,B} I_{k-1}(A + E + B + \{y \in \lambda, : y < x\})$ assigns to z. We can also say that this is the the value which the local theory of x in λ assigns to z. If a, b is an adjacent pair of elements of $I_{k-1}(\{y \in \lambda : y < x\})$, and both $a = (L_0, L_1)$ and $b = (R_0, R_1)$, are in $I_{k-1}(\{y \in \lambda : y < x\}) \cap I_k(\lambda)$, then the value $g((a, b))$ is determined by $I_{(k-2)}(\lambda)$, which names the first and last occurrences of each $k-2$-quantifier type between a and b; the $k-2$-quantifier types realized in (a, b) are, then, those that $I_{(k-2)}(\lambda)$ finds to exist in (a, b). If both $a = (L_0, L_1)$ and $b = (R_0, R_1)$ are in $I_{k-1}(\{y \in \lambda : y < x\}) \cap I_{k-1}(\{y \in \lambda : y < x\}) \cap I_k(\lambda)$, then the value of g is given by $Th^{\text{loc}}_{k-1}(\lambda, x)$. The value of $g(a, b)$ for b an element of $\cap_{A,B} I_{k-1}(A + E + B + \{y \in \lambda : y < x\})$, is determined by the location of x in $I_k(\lambda)$. □

For instance, suppose $I_{(k-2)}(\lambda)$ lists the appearance of quantifier-rank-$k-2$ local types $t_0 \dots t_n$ left of x, and then lists quantifier-rank-$k-3$ local types between these, and x appears between the first appearances of two quantifier-rank-$k-3$ local types, r_0 and r_1, but that r_0 and r_1 both appear between t_{n-1} and t_n. Then, in the gap left of x and all points bound locally to x by lower-rank types, there appear the quantifier-rank-$k-2$ local types $t_0 \dots t_{n-1}$. t_n appears left of x, but only after the disappearance of local type r_1, and not before then (since r_1 disappeared between the last appearances of t_{n-1} and t_n).

Theorem 7.1.1: For any two linear orders, λ, λ', and for any number k, Player II wins $EF_k(\lambda, \lambda')$ iff the following holds: For all sequences $s <^{\text{seq}} (k-1)$, there exists an equivalence $\lambda \equiv_s \lambda'$ – bijections between $I_s(\lambda)$ and $I_s(\lambda')$ identifying the cuts $(A_t^{(a,b)}, B_t^{(a,b)})$ and $(C_t^{(a,b)}, D_t^{(a,b)})$ in both models, for all t and all adjacent pairs a, b of elements in $\cup_{r <^{\text{seq}} s} I_r$ such that whenever B_t has a least element or C_t a greatest element, t of that least element is the same in both models, and such that in every interval in I_k, g has the same value in both models.

Proof (only if): By claim k in lemma 7.1.1, player II must answer player I's first move x_0 with some y_0 in the same interval in I_k. But then, Player II can go on to win only if $Th_{k-1}(\lambda, x_0) = Th_{k-1}(\lambda', y_0)$, so $Th^{\text{loc}}_{k-1}(\lambda, x_0) = Th^{\text{loc}}_{k-1}(\lambda', y_0)$.

Proof (if): If $k = 2$, then $Th_0(\lambda, x) = \{x = x\} = \{t^0\} = Th^{\text{loc}}_0(\lambda, x)$, so $I_2 = I_{(0)}$ describes the first and last elements of type $x = x$, and between them

determines which subset of $Th_1^{\mathrm{loc}}(\lambda, x) = \{x = x\} = \{A{+}E{+}B{+}\lambda{+}C{+}E{+}D \models \exists y(x > y) \wedge \exists y(y > x)\} = \{t^1\}$ is realized. The reader can check that this accurately describes the various $\equiv_2$ classes of linear orders. That the theorem holds for $k \geq 3$ is proved by induction on lemma 7.1.2. □

Remark 7.1.1: The generalization of I_s to infinitary logic (where s is a descending sequence of ordinals) is clear; the lemma that $\equiv_\alpha$ implies $\equiv_s$ goes through as well, except that I_s is no longer finite, and so we must edit the "covering conditions." The restriction of the I_s to the class of ordinals reproduces known results about the theory of ordinals and the infinitary theory of ordinals. For instance, in a finite linear order, for each sequence s, $I_{s'}$ identifies the least and greatest element of the central gap in I_s; the number of nontrivial sequences s of numbers $\leq k-2$ is $2^{k-1}-1$, so the k-quantifier theory of a finite linear orders identifies that many elements on each end of the linear order (If there are any elements of λ not named in that process, they are indistinguishable by formulas of quantifier rank k.

By induction, we build the set of local types $\{\exists x t(x) : t = Th_k^{\mathrm{loc}}(\lambda, x)$ in some model $\lambda\}$ of quantifier-rank k out of structures I_s^L (where we only consider the cuts of the form $(A_t^{(a,b)}, B_t^{(a,b)})$, and not of the form $(C_t^{(a,b)}, D_t^{(a,b)})$), and in turn the structures I_s are built by identifying the first and last elements of each local type. Thus, defining I_s from local types and local types from simplified I_s, we construct the set of local types of the next quantifier rank. Consistency requires that whenever local types t and s are realized and require an element between them which satisfies ϕ, a local formula of lower quantifier rank, then there is some local type r which extends ϕ. We can prove that if these necessary and propositional formulas are satisfied by a set of local types of uniform quantifier rank, then there is a model realizing exactly those local types, via a Model Existence Game (proving that a model can be constructed element by element, and that at no stage is there any obstruction), or by constructing the model at once and in its entirety. The resulting models involve many arbitrary choices about whether to realize s left of t, or t left of s. If we consistently decide to realize both, whenever possible, then the models which are constructed contain intervals which are determined by the total information, and intervals in which all possible types are realized densely.

The inherent complexity of deciding the theory of linear order consists only in the difficulty of extending a formula of quantifier rank k to a total information triple of quantifier rank k, because any total information triple is then easily extended to a complete piecewise-dense model in quantifier rank $k + 2$. By comparison, the proof in [1] is surely the most direct proof that the theory of linear order is atomic, but no reasonable bound on the quantifier rank of the completion can be obtained using that argument. In fact, [4] describes the quantifier rank of the completion obtained by that method, and finds an enormous upper bound, involving towers of exponents in the arguments (page 341, middle paragraph). Studying the propositional theory of sets of local types describes the complexity of the set {total information triples for λ,k : λ is a linear order and k a natural number}. In any case, simply because the theory is propositional, this set is recursively enumerable, thus the theory of linear order is decidable. We have come to that theorem by constructing large finite approximations to a linear order, and I would venture to say that these finite sets are what were described in [2] as the semimodels of a linear order. From

our total information triples, we can construct semimodels for the linear orders – the proof of Lemma 1.1 indicates what elements of the model should be played as corresponding to the various elements of I_s.

7.2 Semimodels and the complexity of deciding $Th(LO)$

A semimodel for the vocabulary $\{<\}$ is, in general, a nested sequence of linear orders $M = (M_0 \subseteq M_2 \subseteq ...M_{k-1}, <^{M_n})$ where we intend to evaluate formulas ϕ of quantifier rank $\leq k$. We write $\phi^{s.m.}$ for the result of replacing the subformula $\exists x\psi$, when it occurs in ϕ within the scope of l_ψ-many quantifiers, by the bounded quantification $\exists x \in M_l\psi$. More precisely,

Definition 7.2.1: A *semimodel* is a nested sequence $M = M_0 \subseteq M_1 \subseteq ...M_{k-1}$ of finite sets, with an ordering on M_{k-1}. The *rank* of M is k.

We say $M \models^{\text{s.m.}} \phi$ if $M \models \phi^{\text{s.m.}}$, where $\phi^{\text{s.m.}}$ is the relativization of ϕ in which any subformula $\exists x_j\psi(x_0 \dots x_j)$ which appears within the scope of j-many quantifiers is replaced by $\exists x_j \in M_j\psi(x_0 \dots x_j)$, for all $j < k$.

Theorem 7.2.1 (Ehrenfeucht [2]): If U is a class of finite semimodels, and the following hold:

1. U is recursively enumerable,
2. For any formula ϕ and any semimodel $L \in U$ such that the rank of L as a semimodel is the rank of ϕ as a formula, if $L \models^{\text{s.m.}} \phi$, then L extends to a full linear order λ, such that $\lambda \models \phi^{s.m.}$,
3. For any formula ϕ and any linear order λ such that $\lambda \models \phi$, $\phi^{\text{s.m.}}$ holds in some semimodel $L \in U$,

then the theory of linear order is decidable.

Proof: Enumerate the implications of the theory of linear order, and look for $\neg\phi$. Meanwhile, enumerate elements of U, and look for $L \in U$ of rank equal to the quantifier rank of ϕ, such that $L \models \phi$. By condition 2., some linear order models ϕ, too. By condition 3, if ϕ is consistent, then this procedure terminates in the discovery of a semimodel of ϕ. □

Definition 7.2.2: If M is a semimodel of rank at least k, and λ is a linear order, we say $M \equiv_k^{\text{s.m.}} \lambda$ if for any sentence ϕ of quantifier rank at most k, $\lambda \models \phi$ just in case $M \models^{\text{s.m.}} \phi$.

We write a semimodel of rank k as a string $s_i : i < n$ of numbers $< k$, from which $M_j = \{i : s_i \leq j\}$ recovers the semimodel.

Given a semimodel M, let M^+ be the semimodel formed by shifting the sets M_i; i.e., $(M^+)_{i+1} = M_i$; $(M^+)_0 = \emptyset$.

Addition of semimodels is the same as addition of orders; if M and N are disjoint and of the same rank, form $(M_{k-1}, <) + (N_{k-1}, <)$, and let $(M + N)_i = M_i \cup N_i$ for $i < k$.

Definition 7.2.3: We call M consistent if when $M \models^{\text{s.m.}} \phi$ then from ϕ is not inconsistent with the theory of linear order.

Note that when we determine $M \models^{\text{s.m.}} \phi$, we restrict the quantifiers of ϕ to the ranked sets in M, and when we determine whether ϕ is consistent, we do not restrict its quantifiers.

For instance, 01 is not a consistent semimodel, since $01 \models^{\text{s.m.}} LO \wedge (\forall x \exists y(y > x)) \wedge (\forall x \neg \exists y(y < x))$, where LO is the theory of linear order. Of course, that is not consistent.

Theorem 7.2.2: For any model M and sentence ϕ satisfied in M, there is a semimodel N such that $N \models^{\text{s.m.}} \phi$.

Proof: To determine that $M \models \phi$, it is never necessary to play more than finitely many elements of M and M^+. To be precise about which elements are played was the work of the previous section, where we found that total information is a finite set, sufficient for determining the k-quantifier theory of M. Turning a total information triple into a semimodel can be done automatically: The elements of M which appear there should be simply added to the semimodel; elements of M^+ which appear in the total information triple represent families of types which are realized cofinally often; if that family of types is codified in the semimodel M, then M^+MM is a semimodel in which each of the types in the family is realized left of any other. □

If that sketch of how to write a string from total information triple seems unsatisfying, I suspect that the problem is mostly that total information triples are complicated objects, which obviously are not necessary in so simple a theorem. Below, we will re-prove this theorem, by constructing semimodels for the linear orders used in [3].

As a result of this theorem, deciding the theory of linear order can be done by deciding which strings are consistent semimodels for the theory of linear order.

Theorem 7.2.3: $(0)^{2^{k-1}}(1)^{2^{k-2}}(2)^{2^{k-3}}...(k-1)^{2^0} \models^{\text{s.m.}} Th_k(\omega)$.

Proof: We play the EF game between a semimodel and a model, where in the semimodel the l-th move is restricted to M_l. We answer a large natural number with the last 0. To the right, this leaves the statement of the theorem for $k-1$, so we appeal to induction. To the left, this leaves $0^{2^{k-1}-1}$, whihc is equivalent to any large, finite linear order, as the reader my check, again by induction. □

We can also see this proof in terms of total information: For each descending sequence s of numbers $< k-1$, I_s contains an element right of those definable in I_r, where r is the predecessor of s. Any of these could be played on the first move. This accounts for $2^{k-1}-1$-many elements on the left end of both ω and the semimodel. The next element of ω, corresponding to the last element of M_0 in the semimodel, is an element not definably close to the left end. It's $k-1$-quantifier type is that it is preceded by a large finite linear order of unknown size, and followed by something $equiv^{k-1}\omega$.

Theorem 7.2.4: If $E \equiv_k^{\text{s.m.}} \eta$, the dense linear order without endpoints, then $E^+ + (0) + E^+ \equiv_{k+1}^{\text{s.m.}} \eta$. $101 \equiv_2^{\text{s.m.}} \eta$; $2120212 \equiv_3^{\text{s.m.}} \eta$.

Proof: Like E, η has only one type (of quantifier rank k) $t(x)$, and after playing x to any element of η, the k-quantifier theory of $\eta^{<x}$ or $\eta^{>x}$ is the k-quantifier theory of η. Base case: $\emptyset \equiv_0^{\text{s.m.}} \eta$. □

Definition 7.2.4: In the game $EF_k^{\text{s.m.}\cdot}(L,\lambda)$ played on a semimodel and a model, player I plays in L or λ, and player II replies in the other structure. On the $j+1$-th move, the player who assigns x_j to an element of L must assign x_j to an element of L_j.

Given two semimodels M and N, call $M \times N$ the semimodel obtained by replacing every element $x \in N$ such that $x \in N_j \setminus N_{j-1}$ by a copy of $M^{(+)^j}$.

Note that player II has a winning strategy in $EF_k^{\text{s.m.}\cdot}(L,\lambda)$ just in case $L \equiv_k^{\text{s.m.}\cdot} \lambda$.

Theorem 7.2.5: If $M \equiv_k^{\text{s.m.}\cdot} \mu$ and $L \equiv_k^{\text{s.m.}\cdot} \lambda$, then $M \times L \equiv_k^{\text{s.m.}\cdot} \mu \times \lambda$.

Proof: Suppose player II has a winning strategy in $EF_k^{\text{s.m.}\cdot}(L,\lambda)$ and in $EF_k^{\text{s.m.}\cdot}(M,\mu)$. On the j-th move of $EF_k^{\text{s.m.}\cdot}(M \times L, \mu \times \lambda)$, if player I has moved in the same element of λ or L as in the previous move, player II follows the winning strategy of $EF_k^{\text{s.m.}\cdot}(M,\mu)$ in that copy of M. If player I plays in a new element of λ or L, player II responds as in $EF_k^{\text{s.m.}\cdot}(L,\lambda)$, and in the corresponding copy of M and μ, begins a new game of $EF_k^{\text{s.m.}\cdot}(M,\mu)$. If player I plays in $\lambda \times \mu$, then player II has answered with an element of $(M \times L)_i$. □

Definition 7.2.5: ([3]) M_{LL} is the smallest set of linear orders containing the singleton order, 1 and closed under the operations

- $+$, where $M + N$ forms disjoint copies of M and N, and then orders all the elements of M before any element of N.
- $\times\omega$, where $M \times \omega$ contains countably many copies of M, indexed by ω, and if $n < m \in \omega$, all the elements of M_n occur before any element of M_m.
- $\times\omega^*$, which has the same domain as $M \times \omega$, but if $n < m$, then in $M \times \omega^*$ all elements of M_m precede any element of M_n.
- σ: Let η be the ordering on the rationals, and let χ be any function from η to n such that $\chi^{-1}(i)$ is dense for each $i < n$. The domain of $\sigma\{M_i : i < n\}$ contains a copy M_q of M_i for each rational number q such that $\chi(q) = i$ – and M_q and M_p are disjoint if $p \neq q$ – and if $p < q$ then every element of M_p precedes any element of M_q.

If we understand that M^* is the ordering with the domain of M in which $a <^{M^*} b$ if $a >^M b$, then we could define $M \times \omega^* = (M^* \times \omega)^*$.

Definition 7.2.6: Let En be the semimodel such that $En_j = (En_{j-1}^+ + (0)) \times n + En_{j-1}^+$. Let χ map En to n, sending the i-th occurrence of 0 in the construction of En_j to i, for all j. Let $\sigma(\{M_i : i < n\})$ be the semimodel obtained from En by replacing every element $x \in En$ such that $\chi(x) = i$ and $x \in N_j \setminus N_{j-1}$ by a copy of $M_i^{(+)^j}$.

Theorem 7.2.6: $En_k \equiv_k^{\text{s.m.}\cdot} \eta$. $\chi^{-1}(i) \equiv_k^{\text{s.m.}\cdot} \eta$. Let En, χ, and $\sigma(\{M_i : i < n\})$ be as in the preceding definition. If $M_i \equiv_k^{\text{s.m.}\cdot} \mu_i$, then the shuffle of the semimodels is $\equiv_k^{\text{s.m.}\cdot}$ to the shuffle of the linear orders: $\sigma(\{M_i : i < n\}) \equiv_k^{\text{s.m.}\cdot} \sigma(\{\mu_i : i < n\})$.

Proof: As in the proof of the product, Player II follows $E, \chi \equiv_k^{\text{s.m.}} \eta, \chi$, and begins a new game of $EF_k^{\text{s.m.}}(M_i, \mu_i)$ whenever whenever player I changes from one element of E or η to another. Player II follows $EF_k^{\text{s.m.}}(M_i, \mu_i)$, whenever player I keeps the element of E or η constant. □

We can formulate semimodels of rank k for elements of M_{LL} as follows:

- $1 \equiv^{\text{s.m.}} 1$,
- $M + L \equiv_k^{\text{s.m.}} \mu + \lambda$ if $M \equiv_k^{\text{s.m.}} \mu$ and $L \equiv^{\text{s.m.}} \lambda$,
- $M \times L \equiv_k^{\text{s.m.}} \mu \times \omega$ if $M \equiv_k^{\text{s.m.}} \mu$ and $L \equiv_k^{\text{s.m.}} \omega$,
- $M \times L \equiv_k^{\text{s.m.}} \mu \times \omega^*$ if $M \equiv_k^{\text{s.m.}} \mu$ and $L \equiv_k^{\text{s.m.}} \omega^*$,
- $\sigma(\{M_i : i < n\}) \equiv_k^{\text{s.m.}} \sigma(\{\mu_i : i < n\})$ if $M_i \equiv_k^{\text{s.m.}} \mu_i$ for each $i < n$.

Now, the set of semimodels M such that $M \equiv_k \mu$ for some $\mu \in M_{LL}$ is clearly recursively enumerable, and clearly each such M extends to a linear order (namely, μ), so conditions 1 and 2 in Ehrenfeucht's theorem are clear. The final, third, condition is what was proved in [3].

What is the complexity of determining whether a string is consistent? We can show that this question can be determined quickly in the length of a string, i.e., in linear time. However, the length of a string which represents, as a semimodel, a random k-quantifier equivalence class, has length tower-of-exponential in k.

References

[1] Amit, R., Shelah, S.: The complete finitely axiomatized theories of order are dense. Israel J. Math., Vol 23 (1976) 200-208

[2] Ehrenfeucht, A.: Decidability of the theory of linear ordering relation, AMS Notices, Vol 6.3.38 (1959) 556-538

[3] Läuchli, H., Leonard, J.: On the elementary theory of linear order, Fund. Math., Vol LIX (1966) 109-116

[4] Rosenstein, J.: Linear Orderings. Academic Press (1982) 341-342

INDEX