Tree representations via ordinal machines

Philipp Schlicht and Benjamin Seyfferth

schlicht@math.uni-bonn.de
seyfferth@math.uni-bonn.de
Mathematical Institute, University of Bonn
Endenicher Allee 60, 53115 Bonn, Germany

Abstract. We study sets of reals computable by ordinal Turing machines with a tape of length the ordinals that are steered by a standard Turing program. The machines halt at some ordinal time or diverge. We construct tree representations for ordinal semi-decidable sets of reals from ordinal computations. The aim is to generalize uniformization results to classes of ordinal semi-decidable sets defined by bounds on the halting times of computations. We further briefly examine the jump structure and nondeterminism.

1 Introduction

Ordinal computability studies generalized computability theory by means of classical machine models that operate on ordinals instead of natural numbers. Starting with Joel Hamkins’ and Andy Lewis’ Infinite Time Turing Machines (ITTM) [1], recent years have seen several of those models which provided alternate approaches and new aspects for various ideas from logic, set theory and classical areas of generalized computability theory. With ITTMs, the machine may carry out a transfinite ordinal number of steps while writing 0s and 1s on tapes of length ω. This is achieved by the addition of a limit rule that governs the behavior of the machine at limit times. The 0s and 1s on the ω-long tape are interpreted as subsets of ω (reals). It turns out that the sets of reals semi-decidable by these machines form a subset of $\Delta^1_2$. Similar studies have been carried out for infinite time register machines (ITRMs), whose computable reals are exactly the reals in $L_{\omega^\omega}^\omega$ [7].

Another direction of ordinal computability lifts classical computability to study not the subsets of ω, but of an arbitrary ordinal $\alpha$, or even the class Ord of all ordinals. In this case, both space and time are set to that ordinal $\alpha$, i.e. in the Turing context, we deal with machines that utilize a tape of length $\alpha$ and either stop in less than $\alpha$ many steps or diverge. The computation is steered by a standard Turing program and a finite number of ordinal parameters less than $\alpha$. This approach unveils strong connections to Gödel’s universe of constructible sets and the classical work on $\alpha$-recursion theory [8].

In the present paper, we aim between these two approaches by analyzing the computable sets of reals of Turing machines with Ord space and time but without allowing arbitrary ordinal parameters. We work with ordinal Turing machines (OTMs), the machine model introduced in [6]. Let us briefly review the basic features, for more detail and background the reader is referred to the original paper.

An OTM uses the binary alphabet on a one-sided infinite tape whose cells are indexed by ordinal numbers. At any ordinal point in time, the machine head is located on one of these cells and the machine is in one of finitely many machine states indexed by natural numbers. Since we utilize both Ord space and time, there is no need to use multiple tapes in our definition; any fixed finite number of tapes can be simulated by interleaving the tapes into one. A typical program instruction has the form $(a, s, a', s', d) \in \{0, 1\} \times \{0, 1\} \times \{0, 1\} \times \{-1, 1\}$ and is interpreted as the instruction “If the symbol currently read by the machines read-write head is a and the machine is currently in state s, then overwrite a with the symbol a’, change the machine state to s’, and move the head according to d either to the left or to the right”. At successor times in the course of the computation, the machine behaves like a standard Turing machine, with the following exception: If the machine head rests on a cell indexed by a limit ordinal or 0 and a “move left”-instruction is carried out, then the head is set to position 0. The machine accesses the transfinite by the following lim inf-rule:
At a limit time $\lambda$, the machine state is set to the lim inf of the states of previous time, i.e., the least state that was assumed cofinally often in the previous steps. Similarly, we set the tape content for each cell individually to the lim inf of the previous cell contents; in other words, a cell contains a 0 at time $\lambda$ if it contained a 0 cofinally often before $\lambda$, and it contains a 1 at time $\lambda$ otherwise. It is natural to set the head position to the cell indexed by the lim inf over the indices of the cells visited at previous steps in which the machine’s state was the same as in the limit.

These ordinal machines may be used to describe sets of reals. In order to input a real or set of ordinals into an ordinal Turing machine, we start the computation with an initial tape content coding the real or set of ordinals; for a real the initial tape contents is a sequence of numbers into an ordinal Turing machine, we start the computation with an initial tape content coding the real.

Let us denote the OTM computation by a program $P$ on input $x$ as $P(x)$ and abbreviate the statement "$P(x)$ halts" as $P(x) \downarrow$.

**Definition 1.** A set of reals $A \subseteq \omega$ is called OTM semi-decidable if there is an ordinal Turing machine that halts if and only if the initial tape content was an element of $A$. $A$ is called OTM decidable if its characteristic function $\chi_A$ is an OTM computable function.

Our motivation is to use ordinal machines to refine uniformization results in descriptive set theory. Many results in descriptive set theory have simple proofs using admissible sets [3]; we go further than [3] in providing explicit algorithms for the constructions. In section 2, we define an algorithm for searching for infinite branches in the Shoenfield tree, to prove that the $\Sigma_2^1$ sets of reals are exactly the OTM semi-decidable sets of reals. As a consequence, we re-establish Shoenfield’s absoluteness from the perspective of ordinal computability. The fact that the $\Sigma_2^1$ sets of reals are exactly the OTM semi-decidable sets of reals may be alternatively obtained from $\Sigma_2^1$ absoluteness and the fact that bounded truth in $L$ is an OTM computable relation (for the latter see [6]). In section 3, we introduce a tree representation for $\Sigma_2^1$ sets that is based on finite fragments of OTM computations. The main result of this paper is the application of this representation and the algorithm in section 2 to prove uniformization for classes of OTM semi-decidable sets of reals defined by upper bounds on the halting times of computations. Section 4 introduces a notion of nondeterministic OTM computations. Applying our algorithm from section 2 to the tree representation establishes that nondeterministically OTM decidable sets are already deterministically so. We then show that the jump structure or our machines depends on set theoretic assumptions. Let us refer to [4] and [5] for the set theoretic background.

## 2 Computing the Shoenfield tree

In this section, we define an OTM algorithm searching for branches in the Shoenfield tree. For technical reasons, let us fix the following OTM computable functions. The Goedel pairing function is a bijection $(\cdot,\cdot) : \text{Ord} \times \text{Ord} \rightarrow \text{Ord}$. Elements of Baire space can be represented as subsets of $\omega$ by coding their graph via Goedel pairing. The function $o : \omega \rightarrow \omega$ is a computable bijection providing a computable enumeration of the basic open sets $O(i)$ of the Baire space $\omega$, where $O(i)$ denotes the basic open set defined by the sequence $o(i)$.

We will make use of the standard tree representation for $\Pi^1_1$ sets due to Luzin and Sierpinski. Recall that a set $B \subseteq \omega$ is $\Pi^1_1$ if there is a tree $T$ on $\omega$ such that $x \in B$ if and only if $T_x$ is well-founded and the relation $\{(x,i) \mid o(i) \in T_x\}$ is computable. Let us call $T$ the Luzin-Sierpinski tree for $B$. The tree $T_x$ is well-founded if and only if there is an order-preserving embedding of $T_x$ into some countable ordinal $\alpha$. To check whether $x \in B$, we can look for a suitable infinite branch in the tree $S$ on $\omega$ of all pairs $(s,u)$ with $s \in \omega$ and $u \in \omega$ for some $n \in \omega$ where $u$ codes an order-preserving map $f_u : T_x \cap \{ o(i) \mid i < \text{length}(u) \} \rightarrow \omega$. This is the Shoenfield tree projecting to $B$.

Let us first define an algorithm searching the Shoenfield tree for a $\Pi^1_1$ set $B \subseteq \omega$. Let $T$ be the Luzin-Sierpinski tree for $B$. The algorithm shall halt on input $x \in \omega$ if and only if $x \in B$. Depth-first-search
(DFS) is employed to find an infinite branch in the subtree of \( S \) that consists of the pairs \((s,u)\), where \( s = x \upharpoonright n \) for some \( n \in \omega \). In other words, we will search \( S_x \), which is a tree on \( \omega_1 \). Clearly, membership in \( S \) of any given pair \((s,u)\) is OTM decidable in every admissible set, as the property \( p(i) \in T_x \) is computable. Note that \( \omega \) may be used as a constant, since the constant function with value \( \omega \) is OTM computable.

**Algorithm 1**

```plaintext
set \( \alpha = 0 \)

MAIN:
set \( u = () \);
set \( n = 0 \);
call DFS(u);
\( \alpha + \tau \);
call MAIN;

DFS(u):
if \( n = \omega \) then stop;
if \( (x \upharpoonright n, u) \in S \) then set \( n + + \) and set \( u = u \upharpoonright 0 \) and call DFS(u) and set \( n -- \) and set
\( u = u \upharpoonright n \);
if \( u(n) < \alpha \) set \( u(n) + + \) and call DFS(u);
```

The algorithm starts with the empty sequence \( u = () \) and in stage \( \alpha = 0 \). Whenever DFS\((u)\) is called, all possible extensions of \( u \) by a single ordinal \( \beta < \alpha \) are tried. When all \( \beta < \alpha \) have been tried, DFS\((u)\) ends. If a feasible extension \( u \upharpoonright \beta \in S_x \) is found, the recursion will immediately try to extend it further and DFS\((u \upharpoonright \beta)\) is called. Whenever the algorithm tries an extension \( u \upharpoonright \beta \) that is not in \( S_x \), this extension is not followed further and \( u \upharpoonright (\beta + 1) \) is tried next. If the length \( n \) of \( u \) has reached \( \omega \), a branch is found, i.e., \( u \) codes an order preserving embedding of \( T_x \) into the ordinal \( \alpha \). If no branch can be found, the recursion eventually breaks down, \( \alpha \) is incremented, and the algorithm starts over with the empty sequence.

Throughout the algorithm, the variable \( u \) is stored in an extra tape whose \( n \)-th cell contains a 1 if and only if \( n \leq (p, q) \) and \( u(p) = q \) and 0 otherwise. Therefore, the variable also contains the desired value at limit times.

**Lemma 1.** The algorithm will find the lexicographically least infinite branch through \( S_x \), if there is one.

**Proof.** It is clear that if the algorithm finds a branch, it will find the lexicographically least. So we have to show that this branch is eventually found. Let \( v \in {}^\omega \omega_1 \) be the lexicographically least branch of \( S_x \) and let \( \gamma \) be the supremum of the ordinals in \( v \). The tree \( S_x \cap {}^\omega \omega \gamma \) is countable. Observe that the algorithm visits exactly the nodes of \( S_x \cap {}^\omega \omega \gamma \) in the stages \( \alpha < \gamma \) and that every node is visited only once. Since this subtree contains no branches, the algorithm sets \( \alpha = \gamma \) after countably many steps. Note that in stage \( \gamma \), the algorithm will first visit the countably many sequences \( \omega \in S_x \) that are lexicographically smaller than \( v \upharpoonright \text{length}(w) \). No node \( w \) that is lexicographically greater than \( v \upharpoonright \text{length}(u) \) is visited before the algorithm examines every initial segment of \( v \), so the algorithm eventually finds the branch in countable time.

Now consider a \( \Sigma^1_2 \) set \( A \subseteq {}^\omega \omega \) and a \( \Pi^1_1 \) set \( B \subseteq {}^\omega \omega \times {}^\omega \omega \) such that \( p(B) = A \). We will modify the above algorithm to semi-decide the set \( A \). Let \( (T \subseteq {}^2 \omega \times \omega) \) be the Luzin-Sierpiński tree for \( B \). The Shoenfield tree \( S \) for \( B \) is the tree of all \((s,t,u)\) where \( u \) codes an order-preserving embedding \( f_u : T_{x,t} \to \text{Ord} \). Since \( B = p([S]) \) and \( A = p(B) \), we have \( x \in A \) if and only if the tree \( S_x \) (on \( \omega \times \omega_1 \)) has an infinite branch. In order to find such a branch for a given \( x \), the algorithm proceeds in stages \( \alpha \in \text{Ord} \). In each stage \( \alpha \), depth-first-search is employed to find an infinite branch in the subtree of \( S_x \), which consists of the pairs \((t,u)\) where \( t \in \text{length}(u)^\alpha \).
Algorithm 2

\[
\text{set } \alpha = 0;
\]

\text{MAIN:}
\[
\text{set } t = ();
\]
\[
\text{set } u = ();
\]
\[
\text{set } n = 0;
\]
\[
\text{call } \text{DFS}(t, u);
\]
\[
\text{set } \alpha + +;
\]
\[
\text{call } \text{MAIN};
\]

\[
\text{DFS}(t, u):
\]
\[
\text{if } n = \omega \text{ then stop;}
\]
\[
\text{if } u(n) = \alpha \text{ then set } u(n) = 0 \text{ and set } t(u) + +;
\]
\[
\text{if } t(n) = \omega \text{ then set } n \leftarrow \text{ and set } t = t \upharpoonright n \text{ and set } u = u \upharpoonright n;
\]
\[
\text{if } (x \upharpoonright n, t, u) \in S \text{ then set } n + + \text{ and set } t = t \upharpoonright 0 \text{ and set } u = u \upharpoonright 0 \text{ and call } \text{DFS}(t, u) \text{ and set } n \leftarrow \text{ and set } t = t \upharpoonright n \text{ and set } u = u \upharpoonright n;
\]
\[
\text{set } u(n) + + \text{ and call } \text{DFS}(t, u);
\]

Here in every call of \text{DFS}(t, u), the algorithm tries to extend \( t \) and \( u \) simultaneously by all pairs \((m, \beta)\) with \( m \in \omega \) and \( \beta < \alpha \). Again, if \((t \sim m, u \sim \beta) \in S_x\), the sequence is immediately extended further, i.e. \(\text{DFS}(t \sim m, u \sim \beta)\) is called. Otherwise, \((t \sim m, u \sim \beta + 1)\) is tried next. If for all \( \beta < \alpha \) \((t \sim m, u \sim \beta)\) cannot be extended further, then \((t \sim m + 1, u \sim 0)\) is tried next, and so on.

**Lemma 2.** The algorithm will find the lexicographically least \( z \) such that \( S_x, z \) has a branch, and the lexicographically least branch \( v \) through \( S_x, z \), if such a real \( z \) exists.

**Proof.** Assume \( z \) and \( v \) are as required. As in Lemma 1, we can see that before stage \( \gamma \) (where \( \gamma \) is the supremum of the range of the embedding coded by \( v \)), only countably many nodes are visited. In stage \( \gamma \), only countably many nodes are visited before the branch \((z, v)\) is found.

It is straightforward to generalize this algorithm to semi-decide \( \Sigma^1_2 \) subsets of \( \mathbb{R} \). From the algorithms, we obtain short proofs of several results in classical descriptive set theory.

**Corollary 1.** Suppose \( M \) is a transitive model of KP with \( \omega_1 \subseteq M \). Then \( \Sigma^1_2 \) relations are absolute between \( M \) and \( \mathcal{V} \).

**Proof.** Since OTM computations are absolute between transitive models of KP (see [6, Lemma 2.6]), so is membership in \( \Sigma^1_2 \) sets.

**Corollary 2.** Every \( \Sigma^1_2 \) binary relation on the reals has a \( \Sigma^1_2 \) uniformization and every \( \Pi^1_2 \) binary relation on the reals has a \( \Pi^1_2 \) uniformization.

**Proof.** Suppose \( A \subseteq \mathbb{R} \times \mathbb{R} \) is a \( \Sigma^1_2 \) set. The algorithm semi-deciding \((x, y) \in A\) can be modified to search for a \( y \) given \( x \) as input. As we added the search for sequences \( t \in \mathbb{R} \) to Algorithm 1 to obtain Algorithm 2, we may also add another search for \( s \in \mathbb{R} \) with \((s, t, u) \in S_x\). An argument analogous to Lemmas 1 and 2 proves that the lexicographically least branch \((y, z, v)\) through \( S_x \) is found. This corresponds to the lexicographically least branch through \( S_{x,y} \), therefore \((x, y) \in A\). For any \( \Pi^1_2 \) binary relation, a similar modification of Algorithm 1 yields an algorithm semi-deciding a uniformization such that for any pair \((x, y)\) in the uniformizing function, the algorithm halts before the least \((x, y)\)-admissible \( \omega^\times_z \) above \( \omega \). Hence the uniformization is \( \Pi^1_2 \) by the Spector-Gandy Theorem.

This immediately implies

**Corollary 3.** Every nonempty \( \Sigma^1_2 \) set of reals has a \( \Sigma^1_2 \) member, i.e. some \( x \) such that \( \{x\} \) is a \( \Sigma^1_2 \) set, and every nonempty \( \Pi^1_2 \) set of reals has a \( \Pi^1_2 \) member.
The proof of Corollary 2 shows that any function from the reals to the reals with OTM semi-decidable graph is OTM computable. Note that this is false when we consider OTM programs $P$ such that $P(x)$ halts before $\omega_1^2$ for all $x$ with $P(x) \downarrow$. Let us consider a $\Pi^1_2$ function, obtained via $\Pi^1_2$ uniformization $f$, mapping a real $x$ to a code for a wellfounded countable model containing $x$ of the theory $T$, where $T$ is the extension $\mathbb{KP}$ requiring that there is an admissible ordinal. Although its graph is semi-decidable by such a program, it is easy to see that $f$ is not OTM computable by a program of this type.

Corollary 4. Every $\Sigma^1_2$ set is the union of $\omega_1$ many Borel sets.

Proof. Given a $\Sigma^1_2$ set $A$, let $P$ be an OTM which terminates on input $x$ if and only if $x \in A$. Let $A_\beta$ denote the set of reals $x$ such that $P(x)$ terminates before stage $\beta$. Then $A$ is the union of the sets $A_\beta$.

To see that each $A_\beta$ is Borel, let $a_\beta$ be a real coding the supremum $\gamma_\beta$ over the halting times of the algorithm if restricted to at most $\beta$ stages.\(^1\) Then a real $x$ is an element of $A_\beta$ if and only if for some (for every) real $c$ coding a computation along $a_\beta$, this computation halts. This shows that $A_\beta$ is $\Delta^1_1$ and hence Borel by Suslin’s Theorem.

Corollary 5. Every $\Sigma^1_2$ set has a $\Sigma^1_2$ norm.

Proof. Let $A$ be a $\Sigma^1_2$ set and let $P_A$ be an algorithm semi-deciding $A$. The desired norm is given by the map $\phi$ where $P_A$ halts at time $\phi(x)$ on input $x$. Let $x \leq y$ ($x < y$) if $P(x)$ halts (strictly) before $P(y)$, or $P(y)$ does not halt. Then $y \in A$ and $x \leq y$ imply $x \in A$. Using the algorithm, it is easy to see that the relations $\leq$ and $<$ are OTM semi-decidable. Hence $\phi$ is a $\Sigma^1_2$ norm on $A$.

Note that we cannot obtain a $\Sigma^1_2$ norm whose initial segments are uniformly Borel. This would imply the existence of an uncountable sequence of distinct Borel sets of bounded rank, however this does not follow from $\mathsf{ZF}$ [2, Theorem 4.5].

In order to describe the supremum of the ordinals appearing as the halting time of some OTM program, let $\delta^2_1$ denote the supremum of lengths of $\Delta^1_2$ wellorders on sets of natural numbers. Let $\delta^2_1(x)$ denote the supremum of the length of $\Delta^1_2$ wellorders in the parameter $x$ on sets of natural numbers. Note that a real $x$ is $\Delta^1_2$ if and only if $\{x\}$ is $\Delta^1_2$ or even just $\Sigma^1_2$.

Corollary 6. The supremum of halting times of OTMs with input $x$ is $\delta^2_1(x)$.

Proof. Suppose $y$ codes a $\Delta^1_2$ wellorder in the parameter $x$ of type $\gamma$. Since we may assume that $y$ is OTM computable, consider the algorithm searching for the next element in the wellorder. The algorithm halts as soon as every natural number has appeared at a time at least $\gamma$. If $P$ is a program, let us consider the $\Pi^1_2$ set in the parameter $x$ of pairs $(y, z)$ such that $y$ codes a wellorder $w$ with a maximal element $l$ and domain the natural numbers and $z$ codes a halting computation along $w$ on input $x$ which halts at $l$. This set contains a $\Pi^1_2$ singleton $(y, z)$ in the parameter $x$ by Corollary 2. Then $y$ codes a $\Delta^1_2$ wellorder in the parameter $x$ whose order type is the length of the computation.

As an example of a $\Sigma^1_2$ wellorder of length $\delta^2_1$, let $m <_\text{halt} n$ if $P_m$ and $P_n$ both halt on empty input and $P_m$ halts before $P_n$, or $P_m$ and $P_n$ halt simultaneously and $m < n$.

3 Tree representations from computations

In this section, we construct a tree representation for an OTM semi-decidable set of reals from finite fragments of OTM computations. The tape content over an entire halting OTM computation on countable input by a program $P$ can be viewed as an $\omega_1 \times \omega_1$ matrix filled with zeroes and ones. Every row represents the tape content at a given time. If we add a state and a head position per row, the computation is entirely captured in the resulting diagram:

\(^1\) If the algorithm terminates in stage $\beta$, the machine halts after at most $(\omega^\omega \cdot \beta^\omega) \cdot \beta$ many steps.
We will approximate similar diagrams by adding single bits of information. A **tape bit** \((\alpha, \beta, c, \lambda)\) will consist of:

1. a coordinate \((\alpha, \beta)\) in the \(\omega_1 \times \omega_1\) matrix representing time \(\alpha\) and tape cell \(\beta\)
2. the cell content \(c \in \{0, 1\}\)
3. a countable limit ordinal (or zero) \(\lambda\) – this number will be used to control the limit behavior.

Per row we also need a **machine bit** \([\alpha, s, \gamma, \lambda]\) containing the following information:

1. some time \(\alpha\)
2. a machine state \(s\) of \(P\)
3. a head position \(\gamma \in \omega_1\)
4. a countable limit ordinal (or zero) \(\lambda\) – this number will be used to control the limit behavior.

A finite set of tape and machine bits can be coded into a countable ordinal; fix such a coding. We will now define the tree \(T\) on \(\omega \times \omega_1\): A pair \((s, u) \in \omega \times \omega_1\) is in \(T\) if and only if

1. The set coded by \(u_i\) contains the bits coded by \(u_i\) for \(0 \leq i \leq j < n\) and natural numbers \(c_i\) for \(i < j\) deciding which bits belong to \(u_i\). The remaining elements of the set coded by \(u_i\) are exactly the bits required by the following conditions.
2. Every \(u_i\) contains at most one machine bit for each \(\alpha\) and at most one machine bit and for every pair of \(\alpha\) and \(\beta\).
3. For every tape bit \((0, n, c, \cdot)\) of \(u_i\) with \(i \geq n\), we have that \(c = s_n\), i.e. \(s\) serves as initial segment of the initial tape contents of the partial computation.
4. \(u_0\) contains the machine bit \([0, 0, \cdot, \cdot]\) and the tape bit \((0, 0, 0, \cdot, \cdot)\). Also it contains a machine bit \([\alpha, s, \gamma, \cdot, \cdot]\) plus a tape bit \((\alpha, \gamma, c, \cdot)\) where \(P\) does not contain an instruction for the situation \((c, s)\), i.e. \(\alpha\) is a halting time. So the beginning and the end of the partial computation are fixed.
5. As soon as we have information about a tape cell at time \(\alpha\), we also know the machine state and head position: If \(u_i\) contains a tape bit \((\alpha, \cdot, \cdot, \cdot)\), it also contains a machine bit \([\alpha, \cdot, \cdot, \cdot]\).
6. We always know which tape contents is read by the read-write head: If \(u_i\) contains a machine bit \([\alpha, \cdot, \gamma, \cdot]\), it contains a tape bit \([\alpha, \gamma, \cdot, \cdot]\).
7. If \(u_i\) contains a tape bit \((\alpha, \beta, c, \cdot)\), \(u_{i+1}\) contains bits immediately above and below along the time axis: Let \([\alpha, s, \gamma, \cdot]\) be the corresponding machine bit. If \(\beta = \gamma\) we require \(u_{i+1}\) to contain a tape bit \((\alpha + 1, \gamma, \cdot, \cdot)\) and a machine bit \([\alpha + 1, \cdot, \cdot, \cdot]\) as required by the program \(P\). If \(\alpha\) is a successor, we also similarly require the tape and machine bit of the form \([\alpha - 1, \cdot, \cdot, \cdot]\) that \(P\) implies. Except for those tape bits, all the other tape cells should not change their content, so we add tape bits \((\alpha + 1, \beta, c, \cdot)\) if \(\beta \neq \gamma\). Again, if \(\alpha\) is a successor, we add such tape bits \((\alpha - 1, \beta, c, \cdot)\) for all \(\beta\) but the one for which we already added such a bit according to \(P\).
8. For tape bits of limit times we have to ensure that the tape contents are inferior limits over earlier times: If \( \lambda \) is a limit ordinal, and \((\lambda, \beta, c, \gamma)\) is a tape bit of \( u_i \). Suppose \( c = 0 \). Then there is a tape bit \((\alpha, \beta, 0, \cdot)\) with \( \alpha < \lambda \) in \( u_{i+1} \) and \( \alpha > \alpha' \) for all bits \((\alpha', \beta, \cdot, \cdot)\) in \( u_i \) with \( \alpha' < \lambda \). If \( c = 1 \) then there is a tape bit \((\alpha, \beta, 1, \lambda)\) in \( u_{i+1} \) with \( \alpha < \lambda \) and where \( \alpha \) is larger than any time of a similar bit in \( u_i \). Let \( \alpha' \) be minimal such that \( u_{i+1} \) contains a tape bit of the form \((\alpha', \beta, 1, \lambda)\). Then every tape bit for tape cell \( \beta \) and time \( \bar{\alpha} \) between \( \alpha \) and \( \lambda \) in \( u_{i+1} \) must be of the form \((\bar{\alpha}, \beta, 1, \cdot)\).

9. We also want the machine state at limit times to be a limit inf: If \( \lambda \) is a limit ordinal and \( u_i \) contains a machine bit \([\lambda, s, \cdot, \cdot]\), \( u_{i+1} \) contains a machine bit \([\alpha, s, \cdot, \cdot]\) where \( \alpha < \lambda \) and where \( \alpha \) is larger than any time of a similar bit in \( u_i \). Let \( \alpha' \) be minimal such that \( u_{i+1} \) contains a machine bit of the form \([\alpha', s, \cdot, \cdot]\). Then every machine bit for time \( \bar{\alpha} \) between \( \alpha \) and \( \lambda \) in \( u_{i+1} \) must be of the form \([\bar{\alpha}, s, \cdot, \cdot]\) where \( \bar{s}' \geq s \).

10. We also want to make the head position at limit times a limit inf as in the definition of OTMs. If \( \lambda \) is a limit ordinal, then for every machine bit \([\lambda, s, \gamma, \cdot]\) of \( u_i \), one of the following conditions hold: Either there is a machine bit \([\alpha, s, \gamma, \cdot]\) in \( u_{i+1} \) with \( \alpha < \lambda \) where \( \alpha \) is larger than any time of a similar bit in \( u_i \) and for every machine bit in \( u_{i+1} \) of the form \([\alpha', s, \gamma', \cdot]\) where \( \alpha' \) is between \( \alpha \) and \( \gamma \) we have \( \gamma' \geq \gamma \). Or, alternatively, \( u_{i+1} \) does not contain a bit of the form \([\alpha, s, \gamma, \lambda]\), then we require that there is a bit \([\alpha, s, \gamma', \lambda]\) in \( u_{i+1} \) that is not in \( u_i \) where \( \gamma' < \gamma \) and \( \gamma' \) is greater or equal to any \( \gamma'' < \gamma \) in any bit of \( u_{i+1} \).

This means that every entry of the matrix given by \( u_i \) is extended both up- and downwards along the time axis in \( u_{i+1} \) while respecting the behavior of the program \( P \) and the limit rules involved in the definition of OTMs.

Let us, in the following, refer to the set of \( \alpha \) where \( u_i \) contains a bit of the form \((\alpha, \cdot, \cdot, \cdot)\) as \( \text{dom}(u_i) \). Moreover, let \( \text{dom}(u) \) be the set of those \( \alpha \) that occur in some \( u_i \), \( i \in \omega \).

**Lemma 3.** \( T \) projects to the set of reals semi-decided by \( P \).

**Proof.** First let \( x \) be semi-decidable by \( P \), i.e. \( P(x) \downarrow \). We claim that the halting computation \( C \) implies a branch of \( T_x \). Let \((\lambda_i)_{i<\omega}\) be an enumeration of the limit times involved in \( C \). Let \((\lambda_i, s_i, \gamma_i, \cdot)\) be the corresponding machine bits, and \((\lambda_i, \gamma_i, c_i, \cdot)\) the corresponding tape bits according to \( C \), for \( i \in \omega \). We can make sure that \( u_i \) contains both \([\lambda_i, s_i, \gamma_i, \cdot]\) and \((\lambda_i, \gamma_i, c_i, \cdot)\) and tape and machine bits \([\alpha_i, \cdot, \cdot, \cdot]\), \((\alpha, \gamma_i, \cdot)\) with \( \lambda_m < \alpha < \lambda_n \), for any \( m < n \). Let us close \((u_i)_{i<\omega}\) under above rules using bits compatible with \( C \). It is clear that for any two consecutive limits \( \lambda_k \) and \( \lambda_l \), there is some \( u_i \) which contains bits \((\alpha_i, \cdot, \cdot, \cdot)\), \((\alpha, \cdot, \cdot, \cdot)\) with \( \lambda_j < \alpha < \lambda_k \). Since all bits are chosen form \( C \), the gaps between the \( \lambda_i \) can be filled and \((u_i)_{i<\omega}\) forms a branch in \( T_x \).

Now let \((u_i)_{i<\omega}\) be a branch of \( T_x \). We need to prove that the computation \( C \) by \( P \) on input \( x \) halts. Let \((\lambda_i)_{i<\omega}\) be an enumeration of the limits in \( \text{dom}(u) \).

**Claim.** The ordinals in \( \text{dom}(u) \) are exactly the ordinals \( \lambda_j + n \) for \( j, n \in \omega \).

**Proof (Claim).** By above rules it is clear that every ordinal of the form \( \lambda_j + n \) is in \( \text{dom}(u) \). Suppose that \( \mu + n \in \text{dom}(u_i) \) where \( \mu \neq \lambda_j \) for all \( j \in \omega \). Then it follows from the rules that \( \mu \in \text{dom}(u_{i+n}) \), a contradiction.

The set of bits in \((u_i)_{i<\omega}\) induce a partial matrix \( U \) of the type pictured above. We call a submatrix according to \( P \), if the machine state, head position, and tape contents change only as dictated by \( P \).

**Claim.** For \( \lambda \in (\lambda_i)_{i<\omega} \) the submatrix of \( U \) induced by the rows \( \lambda + n \) for all \( n \in \omega \) is according to \( P \).

**Proof (Claim).** Let \( n \in \omega \) and choose \( i \) minimal such that \( \exists m \lambda + m \in \text{dom}(u_i) \). The rules dictate that \( u_{i+n-m} \) contains unique machine bits for all rows \( \lambda + m, \lambda + m + 1, \ldots, \lambda + n \) and also for all rows between \( \min(\lambda, \lambda + m - n) \) and \( \lambda + m \). Those machine bits and also the tape contents covered by bits present in \( u_i \) are changed only according to \( P \). Of course, new tape cells might have been introduced by tape bits in \( u_j \), \( j > i \). But for any such given tape cell \( \beta \), its content is kept constant except for actions of \( P \).

It remains to show that at limit times, machine state, head positions, and tape contents are inferior limits.
Claim. Let \( \lambda \) be in \( (\lambda_i)_{i<\omega} \). Let \( (\alpha_j)_{j<\omega} \) be an increasing enumeration of \( \text{dom}(u) \cap \lambda \). Then:

(i) For every tape bit \((\lambda, \beta, c, \cdot)\), \( c \) is the inferior limit over the \( d \) in tape bits of the form \((\alpha_j, \beta, d, \cdot)\) in \( \bigcup_{i<\omega} u_i \).

(ii) For every machine bit \([\lambda, s, \cdot, \cdot]\), \( s \) is the inferior limit over the \( r \) in machine bits of the form \([\alpha_j, r, \cdot, \cdot]\) in \( \bigcup_{i<\omega} u_i \).

(iii) For every machine bit \([\lambda, s, \gamma, \cdot]\), \( \gamma \) is the inferior limit over the \( \delta \) in machine bits of the form \([\alpha_j, s, \delta, \cdot]\) in \( \bigcup_{i<\omega} u_i \).

Proof (Claim).

(i) Choose \( u_i \) such that \((\lambda, \beta, c, \cdot)\) is in \( u_i \). Let \((\alpha_k, \beta, d_k, \cdot)\) be an increasing (in \( \alpha_k \)) enumeration of the tape bits in \((u_j)_{i<j<\omega}\) where \( \alpha_j < \lambda \). First consider \( c = 0 \). The rules imply that \((d_k)_{k<\mu}\) contains an unbounded sequence of 0s, hence \( c \) is in fact the inferior limit. Now suppose \( c = 1 \). In \( u_{i+1} \) a tape bit of the form \((\alpha, \beta, 1, \lambda)\) is added and all \( d_k \) where \( \alpha_k > \alpha \) are \( \geq 1 \).

(ii) Choose \( u_i \) such that \([\lambda, s, s, \cdot]\) is in \( u_i \). Let \((\alpha_k, s_k, \cdot, \cdot)\) be an increasing (in \( \alpha_k \)) enumeration of the machine bits in \((u_j)_{i<j<\omega}\) where \( \alpha_j < \lambda \). In \( u_{i+1} \) a machine bit of the form \([\alpha, s, s, \lambda]\) is added, where \( \alpha \) is greater than any time of a similar bit in \( u_{i+1} \). Indeed in every \( u_i \) where \( j > i \) such a bit is added, so \( (s_k)_{k<\mu} \) contains \( s \) unboundedly often. Also, the rules imply that every \( s_k \geq \alpha \) for all \( \alpha \).

(iii) Choose \( u_i \) such that \([\lambda, s, \gamma, \cdot]\) is in \( u_i \). Let \((\alpha_k, s, \gamma, \cdot)\) be an increasing (in \( \alpha_k \)) enumeration of the machine bits in \((u_j)_{i<j<\omega}\) where \( \alpha_j < \lambda \) (note that we only consider bits with machine state \( s \)). Case 1. In \( u_{i+1} \) a machine bit of the form \([\alpha, s, \gamma, \lambda]\) is added, where \( \alpha \) is greater than any time of a similar bit in \( u_{i+1} \). Indeed in every \( u_i \) where \( j > i \) such a bit is added, so \( (\gamma_k)_{k<\mu} \) contains \( \gamma \) unboundedly often. Also, the rules imply that every \( \gamma_k \geq \gamma \) for all \( \alpha_k \geq \alpha \). Case 2. No such bit is added in any \( u_j \), \( i < j \). Then by the rules, \((\gamma_k)_{k<\mu}\) is strictly increasing below \( \gamma \). Note that by the rules there is no head position in \( u \) that is between \( \text{sup}_{k<\mu}(\gamma_k) \) and \( \gamma \). So even if \( \text{lim inf}_{k<\mu}(\gamma_k) < \gamma \), the partial computation behaves as if \( \gamma \) was indeed the \( \text{lim inf} \).

Let us order the set of tape bits and the set of machine bits lexicographically. If \( u = \{u_0 < ... < u_n\} \) and \( v = \{v_0 < ... < v_n\} \) are (codes) for finite sets of tape bits or finite sets of machine bits, let \( u < \text{lex} v \) if \( u \) is an initial segment of \( v \) or \( u_i \text{ lex} v_i \) for the least \( i \) with \( u_i \neq v_i \). Suppose \( u_n, v_n \) are codes for finite sets as in the definition of the tree \( T \) of partial computations. Then we can decode sequences \( u = \{u_0 < ... < u_n\} \) and \( v = \{v_0 < ... < v_n\} \) from \( u_n \) and \( v_n \) such that \( u_{i+1} \) contains exactly the bits necessary to extend \( u \), for all \( i < n \), and similarly for \( v \). Let us define a wellordering of such codes by \( u_n < \text{code} v_n \) if \( u \) is an initial segment of \( v \) or \( u_i \text{ lex} v_i \) for the least \( i \) with \( u_i \neq v_i \).

Lemma 4. \( T \) has pointwise leftmost branches with respect to \( \text{code} \).

Proof. We claim that for every input \( x \) on which the computation halts, the tree \( T_x \) has a branch \( b \) so that \( b_0 \leq \text{code} c_n \) for every branch \( c \) of \( T_x \) and for every \( n \). Let us consider the computation with input \( x \). Let \( b_0 = \{(0, 0, 0, \cdot), (0, 0, 0, \cdot), (\alpha, s, \gamma, \cdot), (\alpha, \gamma, c, \cdot)\} \), where \( \alpha \) is the halting time, \( s \) is the machine state at time \( \alpha \), \( \gamma \) is the head position at time \( \alpha \), and \( c \) is the content of cell \( \gamma \) at time \( \alpha \). Let \( b_{n+1} \) be the \( \text{lex} \)-least extension of \( b_n \) which describes a fragment of the computation as the definition of \( T \). Suppose towards a contradiction that \( c \) is a branch in \( T_x \) and \( n \) is minimal with \( c_n < \text{code} b_n \). We can decode sequences \( u = \{b_0 < ... < b_n\} \) and \( v = \{c_0 < ... < c_n\} \) from \( b_n \) and \( c_n \). Then \( b_i \leq \text{code} c_i \) for all \( i < n \) by minimality and hence \( b_i \leq \text{code} c_i \). Since \( c_n < \text{code} b_n \), \( b_i = c_i \) for all \( i < n \). This contradicts the choice of \( b_n \).

In particular, the tree induces a \( \Sigma_1^1 \) scale on the set semi-decided by \( P \).

A natural question is whether there is a tree \( T \) projecting to a \( \Sigma_1^1 \) universal set \( A \) such that \( T_x \) has a unique infinite branch for every \( x \in A \). Let us argue that the existence of such a tree is not provable in \( \text{ZF} \). Assuming the existence of such a tree, we can easily convert it into a tree \( S \) with the property that there is a unique \( b \in S \) for every \( x \in A \) and \( b \neq b_y \) for all \( x \neq y \) by coding the first coordinate into the second. Since for each \( \alpha < \omega_1 \) the projection of \( S \) restricted to ordinals below \( \alpha \) is an injective image of a closed set and hence Borel, there is an \( n \) such that the set \( B \) of values of \( b_n(x) \) for \( x \in A \) is unbounded in \( \omega \). Let us choose the leftmost branch \((x_n, b_n)\) in \( S \) with \( b_n(n) = \alpha \) for each \( \alpha \in B \). We have defined an uncountable sequence \((x_n : \alpha \in B)\) of distinct reals. However, there is no such sequence in the symmetric forcing extension for the collapse \( \text{Col}(\omega, < \kappa) \) below an inaccessible cardinal \( \kappa \).
The tree representation allows us to generalize the results in section 2 to sets of reals semi-decided by ordinal machines with upper bounds on the halting times.

**Definition 2.** Suppose \( f \) is a function from the reals to the ordinals. We call \( f \) superadditive if \( f(x) \leq f((x,y)) \) for all reals \( x \) and \( y \). Let us call \( f \) admissible if it is superadditive and \( f(x) \) is \( x \)-admissible for all reals \( x \).

In fact, we consider only additive Turing invariant functions.

**Definition 3.** Suppose \( f \) is superadditive. Let us say that a set of reals \( A \) is \( f \)-semi-decidable or \( \Gamma_f \) if there is an OTM program \( P \) semi-deciding \( A \) such that \( P \) halts before time \( f(x) \) on input \( x \) if it halts at all.

The classes \( \Gamma_f \) for admissible \( f \) with values strictly above \( \omega \) range from \( \Pi^1_1 \) to \( \Sigma^1_2 \).

**Lemma 5.** Let \( f(x) = \omega^1_2 \) and \( g(x) = \delta^1_2(x) \) (see section 2). Then the \( \Pi^1_1 \) sets are exactly the \( f \)-semi-decidable sets and the \( \Sigma^1_2 \) sets are exactly the \( g \)-semi-decidable sets.

**Proof.** Suppose that \( A \) is \( f \)-semi-decidable and \( x \) is a real. Then \( x \in A \) if and only if in every countable model of KP, there is a halting computation with input \( x \). Suppose \( A \) is \( \Pi^1_1 \) and \( x \in A \) if and only if \( T_x \) is wellfounded. Then \( \text{rank}(T_x) < \omega^1_2 \) and hence the algorithm searching for a branch in the Shoenfield tree halts before \( \omega^1_2 \). The statement for \( \Sigma^1_2 \) sets follows from Corollary 6.

**Corollary 7.** Suppose \( f \) is superadditive. Then every \( \Gamma_f \) set has a \( \Gamma_f \) norm.

**Proof.** Suppose a set in \( \Gamma_f \) is semi-decidable by a program \( P \) with halting time bounded by \( f \). Let \( \phi(x) \) be the halting time of \( P \) on input \( x \). The superadditivity of \( f \) implies that \( \phi \) is a \( \Gamma_f \)-norm as in the proof of Corollary 5.

**Corollary 8.** Suppose \( f \) is admissible. Then every \( \Gamma_f \) binary relation has a \( \Gamma_f \) uniformization.

**Proof.** Suppose a relation in \( \Gamma_f \) is semi-decidable by a program \( P \) with halting time bounded by \( f \). We apply the algorithm for searching through the Shoenfield tree to the tree of partial computations. Let us consider the algorithm \( Q \) which on input \((x,y)\) searches for a real \( z \) and a branch in the tree of halting computations of \( P \) with input \((x,z)\). The algorithm will find the lexicographically least such pair, if there is any. In this case the computation halts before \( \alpha = f((x,y)) \), since \( \alpha \) is admissible and hence the tree of partial computations of \( P \) with input \((x,y)\) has a branch in \( L_\alpha[x,y] \). If \( y = z \) we let \( Q \) halt and diverge otherwise. Notice that for any real \( x \) in the domain of the relation there is a lexicographically least pair consisting of a real \( g(x) \) and a branch through the tree of partial computations on input \((x,g(x))\). Hence for any \( x \) in the domain there is a real \( z \) with \( Q(x,z) \downarrow \). Since \( Q \) diverges for all inputs \((x,z)\) with \( z \neq g(x) \), we have found a uniformization.

## 4 Non-deterministic ordinal machines, oracles, and jumps

In [6, Definition 1] the programs that steer the computations of OTMs are defined with the following condition: If the machine is currently in state \( s \) and the machine’s read-write head currently reads symbol \( c \), then the program contains at most one command for that situation. This way, when Koepke defines the ordinal computation by a program \( P \), he can refer to the unique command in a given situation. Instead, for the present section, we shall drop the above restriction on programs and instead define ordinal computations in a way that, in successor steps, the lexicographically least instruction (if there is one that suits the current situation) is chosen to determine the next machine step. This allows us to define non-deterministic ordinal computations as follows.

**Definition 4.** Given program \( P \) and an input (i.e. an initial tape configuration), the non-deterministic ordinal Turing computation (NOTM computation) by \( P \) is defined like the ordinal computation by \( P \) ([6, Definition 2]), except that in successor steps any suitable command may define the machine’s next step.
NOTM computations may be used to define sets of reals.

**Definition 5.** A set of reals \( A \subseteq \omega \) is NOTM semi-decidable if there is a program \( P \) such that

\[
x \in A \iff \text{there is a halting NOTM computation by } P \text{ on input } x
\]

By a Mostowski collapse argument, we obtain for every such \( x \in A \) a countable halting NOTM computation by \( P \) on input \( x \). As in the classical case, given a coding of the “choices” that a NOTM computations makes, NOTM decidability can be verified deterministically.

**Lemma 6.** There is a program \( Q \) such that for every program \( P \) and every real input \( x \), there is a real \( z \) such that the OTM computation by \( Q \) on inputs \( P, x, \) and \( z \) halts if and only if the NOTM computation by \( P \) on input \( x \) halts.

**Proof.** Let us define \( z \) to code two reals \( z_1 \) and \( z_2 \). Let \( z_1 \) code a well-order on \( \omega \) of order type the (countable) length of the NOTM computation by \( P \) on input \( x \). Let \( z_2 \) be such that in machine step \( \text{otp}_{z_1}(i) \), the OTM computation by \( P \) on input \( x \) selects the \( z_2(i) \)-th least command \( P \) contains for that situation. Note that both \( \text{otp} \) and \( \text{otp}_{z_1}^{-1} \) are OTM computable functions. Now the program \( Q \) is essentially a universal OTM which selects the \( z_2(i) \)-th command in \( P \) in the \( \text{otp}_{z_1}(i) \)-th simulation step.

An immediate question is whether NOTMs compute more sets of reals than OTMs.

**Proposition 1.** Every NOTM computable set of reals is already OTM computable.

**Proof.** Let \( A \subseteq \omega \) and suppose that \( Q \) is the program from Proposition 6. By Proposition 6, \( A \) is NOTM semi-decidable if and only if there is a program \( P \) so that for every input \( x \) there is a real \( z \) such that the OTM computation by \( Q \) on inputs \( P, x, \) and \( z \) halts. Since this is a \( \Sigma^1_2 \) statement, \( A \) is \( \Sigma^1_2 \) and hence OTM semi-decidable.

The approach of trying out every coding of choices one after the other could fail if for a given \( x \) every certificate \( z \) was non-constructible. Let us search for a certificate via the tree of partial computations; this can alternatively be done by applying Shoenfield absoluteness and searching through \( L \), using the OTM computable recursive truth predicate from [6].

**Lemma 7.** Given a program \( P \) and an element \((s,t)\) of the full tree on \( \omega \times \omega_1 \), we can OTM decide the question whether or not \((s,t)\) is an element of the tree of partial computations according to \( P \).

**Proof.** We first have the OTM check whether \( t \) is of the correct type. If yes, we can easily check the finitely many conditions if \( t \) is a partial computation by \( P \) on some input that is compatible with \( s \).

With the preceding lemma, we can use a variant of Algorithm 2 to find branches in the tree of partial computations. Since propositions 1 and 2 hold also for our algorithm operating on the tree of partial computations, we get:

**Proposition 2.** If \( A \) is NOTM semi-decidable by the program \( P \), then, given \( x \) as an input, the algorithm will find a real \( z \in \omega \) such that the OTM computation by \( Q \) on inputs \( P, x, \) and \( z \) if \( x \in A \) and diverges otherwise.

**Proof.** If \( x \) is in \( A \), there is a \( z \) in \( L \) such that \( Q_{\text{OTM}}(P, x, z) \downarrow \). An argument analogous to propositions 1 and 2 shows that given a real \( x \), the algorithm will find a branch of the form \((x, c)\) in the tree \( T \) of partial computations by \( P \), if any exists. From \( c \) the desired \( z \) can be easily decoded.

Let us now consider ordinal machines with a set of reals as oracle as in [1]. In a query state in a computation, the program asks whether the sequence on the initial segment of length \( \omega \) of the tape is an element of the set. Let us write \( P^A(x) \) for the OTM computation by the program \( P \) with oracle \( A \) on input \( x \). Let us also fix a computable enumeration \( (P_n | n \in \omega) \) of all programs.

**Definition 6.** The halting problem relative to a set of reals \( A \) or jump of \( A \) is defined as \( A^* = \{(n, x) | P^A_n(x) \downarrow \} \).
The halting problem $0^*$ is a $\Sigma^1_2$ set, in fact we have

**Proposition 3.** The halting problem $0^*$ is $\Sigma^1_2$ universal. If $n \geq 1$ and $V = L$, then the iterated jump $0^{*n}$ is $\Sigma^1_{n+1}$ universal.

**Proof.** Every halting computation with countable input halts at a countable time. Hence $(m,x) \in 0^{*n}$ is described by a $\Sigma^1_{n+1}$ formula stating the existence of a wellorder $w$ on the natural numbers with largest element $l$ together with a sequence indexed by $w$, coding a computation of $P_m$ with input $x$ and oracle $0^{*(n-1)}$ halting at $l$. Let us suppose that $A$ is defined by the formula $\exists x \varphi(x,y)$, where $\varphi$ is $\Pi^1_2$. We consider a program searching through $L$ for a witness for $\varphi$ as in [6], using the oracle $0^{*n}$ to verify $\varphi(x)$ for reals $x$. This program identifies $A$ as a section of $0^{*n}$.

In particular, the $\Sigma^1_{n+1}$ sets are exactly the OTM semi-decidable sets in a $\Sigma^1_1$ oracle for $n \geq 1$, if $V = L$. Let us show that this remains true when $\kappa \geq \omega_1$ many Cohen reals are added to $L$ by the forcing $Add(\omega,\kappa)$.

**Lemma 8.** Suppose that $V = L[G]$, where $G$ is $Add(\omega,\kappa)$-generic over $L$, and $\kappa \geq \omega_1$. Then $0^{*n}$ is $\Sigma^1_{n+1}$ universal for all $n \geq 1$.

**Proof.** Suppose that $x$ is a real in $L[G]$. There are an $Add(\omega,1)$-generic filter $g_0$ with $L[x] = L[g_0]$ and an $Add(\omega,\kappa)$-generic filter $g_1$ over $L[g_0]$ with $L[G] = L[g_0][g_1]$. Let us consider a formula $\exists y \varphi(x,y)$ where $\varphi$ is $\Pi^1_2$. Then $L[G] \models \exists y \varphi(x,y)$ holds if and only if $\exists \sigma \in N \models L[g_0] \models \varphi(y,\sigma)$, where $N$ is the set of nice $Add(\omega,1)$-names for reals in $L[g_0]$. Since nice names for reals are coded by reals, this is a $\Sigma^1_1$ statement in $L[g_0]$. In a similar fashion, we can express any $\Sigma^1_{n+1}$ statement about $x$ in $L[G]$ by a $\Sigma^1_{n+1}$ statement in $L[g_0]$ for all $n \geq 1$, uniformly in $x$. Every such set is a section of $0^{*n}$ by the argument in the previous proposition.

It is also consistent with ZFC that the iterated jumps have a lower complexity. Note that the assumption that $\omega_1^{L[x]} < \omega_1$ for every real $x$ may be obtained by forcing with the Levy collapse $Col(\omega,\kappa)$ below an inaccessible cardinal $\kappa$.

**Lemma 9.** Suppose that $\omega_1^{L[x]} < \omega_1$ for every real $x$. Then $0^{*n}$ is a $\Delta^1_3$ set for all $n \geq 1$.

**Proof.** Let us consider a $\Delta^1_3$ set $A$ of pairs $(x,y)$ where $y$ codes $L_\gamma[x]$ and $\gamma$ is the least $x$-admissible ordinal above $\omega_1^{L[x]}$. We compute the truth value of $x \in 0^{*n}$ in $L_\gamma[x]$, where $\gamma$ is the least $x$-admissible above $\omega_1^{L[x]}$, using an algorithm which has access to $n$ distinct tapes of length $\omega_1^{L[x]} + 1$. The original program runs on tape $n$. Whenever the oracle $0^{*n}$ is called on tape $i$, the oracle is computed on tape $i - 1$ by a subroutine of length $\omega_1^{L[x]} + 1$. The $\Delta^1_3$ description of $A$ provides us with a $\Delta^1_3$ description of the set of pairs $(x,n)$ with $x \in 0^{*n}$.

The two lemmas together show that the complexity of $0^{*n}$ for $n \geq 2$ is independent of the size of the continuum.

5 Further questions

A set of reals is $\Sigma^1_2$ in a countable ordinal $\alpha$ if there is a $\Sigma^1_2$ formula $\varphi(x,y)$ such that for all reals $y$ coding $\alpha$ and all reals $x, x \in A$ if and only if $\varphi(x,y)$ holds, i.e. the $\Sigma^1_2$ definition is independent of the coding of $\alpha$. We leave open whether the sets of reals with a $\Sigma^1_2$ definition in an ordinal $\alpha$, evaluated in $V^{Col(\omega,\alpha)}$, are exactly the OTM semi-decidable sets of reals with input $\alpha$. If so, this might be used for a proof of uniformization for these classes.

References