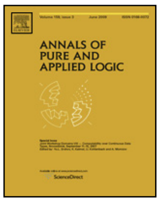
Contents lists available at ScienceDirect

Annals of Pure and Applied Logic

journal homepage: www.elsevier.com/locate/apal



Ordinal machines and admissible recursion theory

Peter Koepke*, Benjamin Seyfferth

University of Bonn, Mathematical Institute, Beringstr. 1, D-53115 Bonn, Germany

ARTICLE INFO

Article history:
Available online 15 February 2009

MSC:
03D60
03D10

Keywords:
Ordinal machines
Computability theory
Admissible recursion theory
α-recursion theory
Sacks–Simpson theorem

ABSTRACT

We generalize standard Turing machines, which work in time ω on a tape of length ω, to α-machines with time α and tape length α, for α some limit ordinal. We show that this provides a simple machine model adequate for classical admissible recursion theory as developed by G. Sacks and his school. For α an admissible ordinal, the basic notions of α-recursive or α-recursively enumerable are equivalent to being computable or computably enumerable by an α-machine, respectively. We emphasize the algorithmic approach to admissible recursion theory by indicating how the proof of the Sacks–Simpson theorem, i.e., the solution of Post's problem in α-recursion theory, could be based on α-machines, without involving constructibility theory.

1. Introduction

Ideas and methods of classical recursion theory have been lifted from the natural numbers to other kinds of mathematical objects. Takeuti [14], Kreisel and Sacks [6], Kripke [7], and Platek [10] defined recursion theory on *ordinals*, making use of the arithmetic and order-theoretic similarities between natural numbers and ordinal numbers. This work led to the concept of *admissible* ordinals and to α*-recursion theory* where recursions are carried out on (the elements of) an admissible ordinal α. The field of α-recursion theory was developed comprehensively by G. Sacks and his school since 1965. Sacks gave the following characterization in his definitive monograph [11, p. 149]:

> α-recursion theory lifts classical recursion theory from ω to an arbitrary Σ_1 admissible ordinal α. Many of the classical results lift to every α by means of recursive approximations and fine structure techniques.

The lifting is based on the observation that a set $A \subseteq \omega$ is recursively enumerable iff it is definable by a Σ_1-formula over $(H_\omega, \in)$, the set of all hereditarily finite sets. By analogy, a set $A \subseteq \alpha$ is said to be α*-recursively enumerable* iff it is $\mathbf{\Sigma}_1(L_\alpha)$, i.e., definable by a Σ_1-formula, allowing parameters, over $(L_\alpha, \in)$ where L_α is the αth level of Gödel's constructible hierarchy. Consequently a set $A \subseteq \alpha$ is said to be α*-recursive* iff it is $\mathbf{\Delta}_1(L_\alpha)$, i.e., if the set and its complement are α-recursively enumerable. So α-recursion theory is closely connected to set theory, in particular to constructibility theory. Its methods involve set-theoretic definability arguments up to the beginnings of Jensen's fine structure theory of the constructible hierarchy. Further information on the connection to set theory can be found in [1].

Besides definability and constructibility techniques, there has always been a strong "computational" attitude in α-recursion theory as described by Sacks [11, p. 155] in the discussion of a $\mathbf{\Sigma}_1(L_\alpha)$-definition (of some function f):

> The definition of f can be thought of as a *process*. At *stage* δ it is assumed that all *activity* at previous stages is encapsulated in an α-finite object, $s \restriction \delta$. In general it will be necessary to *search* through L_α for some existential witness ... [emphases by the present authors].

* Corresponding author.
E-mail address: koepke@math.uni-bonn.de (P. Koepke).

0168-0072/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.apal.2009.01.005

However, no machine model for admissible recursion theory was elaborated. Levy announced a generalization of Turing machines working on regular cardinals in the abstract [8], but no further details were published. Other approaches to ordinal recursion theory were based on recursion schemata (see Takeuti [14], Machover [9]).

In this article we show that α-recursion does indeed correspond naturally to computations by (abstract) machines: recursive enumerability and recursiveness in admissible recursion theory is equivalent to enumerability and computability by certain Turing *machines* working on ordinals. This was proved by the first author (Section 2) after a talk by Sy Friedman on ordinal recursion theory at the Bonn International Workshop on Ordinal Computability (BIWOC) in January 2007. The second author recast the proof of the Sacks–Simpson theorem using the computational paradigm instead of constructibility theory. The crucial point involved was how the informally presented recursions in the argument of Sacks and Simpson [11, 12] (and a recursion method presented by Shore [13]) can be implemented by means of computational mechanisms of the generalized Turing machines (see Section 4). We are very thankful to Russell Miller for his generous help with the discussion of the priority arguments.

Ordinal computability is introduced here on the basis of a Turing machine model, but the theory is robust with respect to various modifications. One could, e.g., use Turing machines with any finite number of tapes and read–write heads, change the commands of the machine, or work with register machines instead of Turing machines.

2. Ordinal Turing machines and the constructible hierarchy

A standard Turing machine is based on the set $\omega = \{0, 1, \ldots\}$ of natural numbers: it acts on a Turing tape of length ω within a discrete time axis which is also indexed by ω. In [5], the first author defined *ordinal* Turing *machines* by replacing the set ω of natural numbers by the class Ord of ordinal numbers. In this article we generalize both standard and ordinal Turing machines to α-Turing *machines*, or α-*machines* for short, where space and time are indexed by (the elements of) some fixed limit ordinal α or by $\alpha = \infty = \mathrm{Ord}$. We define α-machines by an intuitive description of α-*computations*. The relationship between ordinal Turing machines and the constructible model L was studied in [5]. We shall make use of those methods by restricting them to α.

An α-machine possesses a *tape* of length α which consists of *cells* containing the symbols 0 or 1 where 0 is the default symbol. So at every *time* $t < \alpha$ the tape can be formalized by a 0–1-sequence

$$T(t) : \alpha \to 2, \qquad T(t) = (T_0(t), T_1(t), \ldots).$$

A *read–write head* is moving on the tape, positioned at an ordinal $H(t) < \alpha$ at time $t < \alpha$. The head starts at cell 0, i.e., $H(0) = 0$. The computation is steered by a (standard) Turing *program* which is a finite set P of numbered commands of the two forms

```
s: if head=c then print c', move right, and change to state s'
```

or

```
s: if head=c then print c', move left, and change to state s'
```

where c, c'$\in \{0, 1\}$ and s, s'$\in \omega$. At time t the machine is in some *state* $S(t) \in \omega$, starting from state 0, i.e., $S(0) = 0$. A computation of the machine is a sequence

$$(T(t), H(t), S(t))_{t<\theta}$$

of machine configurations $(T(t), H(t), S(t))$ for t below some maximal $\theta \leq \alpha$ at which the computation *stops*. The computation is defined by recursion on $t < \theta$. The *initial configuration* is of the form $(T(0), 0, 0)$.

At *successor times* $t + 1$ the configuration is defined from the configuration $(T(t), H(t), S(t))$ as follows. Let s$= S(t)$, and let c$= T_{H(t)}(t)$ be the symbol under the machine's head.
Case 1. The program contains a command of the form

```
s: if head=c then print c', move right, and change to state s'
```

Then the machine acts accordingly by setting

$$\begin{aligned} T_\xi(t+1) &= \begin{cases} T_\xi(t), \text{if } \xi < \alpha \text{ and } \xi \neq H(t) \\ \texttt{c'}, \text{otherwise} \end{cases} \\ H(t+1) &= H(t) + 1 \\ S(t+1) &= \texttt{s'} \end{aligned}$$

and proceeds to time $t + 1 < \alpha$.
Case 2. Not *Case 1* and the program contains a command of the form

```
s: if head=c then print c', move left, and change to state s'
```

Then set

$$
\begin{aligned}
T_\xi(t+1) &= \begin{cases} T_\xi(t), \text{ if } \xi < \alpha \text{ and } \xi \neq H(t) \\ \texttt{c'}, \text{ otherwise} \end{cases} \\
H(t+1) &= \begin{cases} H(t)-1, \text{ if } H(t) \text{ is a successor ordinal} \\ 0, \text{ otherwise} \end{cases} \\
S(t+1) &= \texttt{s'}
\end{aligned}
$$

and proceed to time $t+1 < \alpha$. Note that if $H(t)$ is a limit ordinal then the head position is reset to 0.
Case 3. Not *Case 1* and not *Case 2*. Then the computation stops, i.e., θ is set to the successor ordinal $t+1$.

The configuration at *limit times* $t < \alpha$ is obtained as a natural limit of previous configurations, using lim inf-operations:

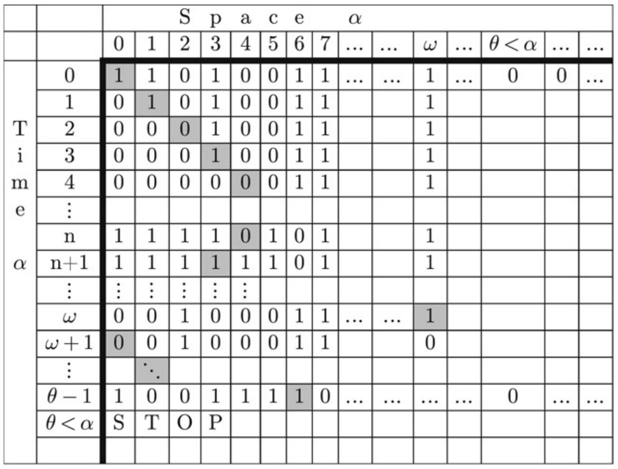
$$
\begin{aligned}
T(t)_\xi &= \liminf_{s \to t} T(s)_\xi \\
S(t) &= \liminf_{s \to t} S(s) \\
H(t) &= \liminf_{s \to t, S(s)=S(t)} H(s).
\end{aligned}
$$

So, if the contents $T(s)_\xi$ of the ξth cell of the tape stabilize before time t then at time t the ξth cell contains that stable value; otherwise $T(t)_\xi$ is set to the default 0. The definitions of $S(t)$ and $H(t)$ can be motivated as follows. Since a Turing program is finite, its execution will lead to some (complex) looping structure with loops, subloops and so forth which can be presented by pseudo-code like:

```
    ...
17:begin loop
       ...
21:   begin subloop
          ...
29:   end subloop
       ...
32:end loop
    ...
```

Assume that for times $s \to t$ the loop $(17-32)$ with its subloop $(21-29)$ is traversed cofinally often. At limit time t it is natural to put the machine back to the beginning (17) of the "main loop". Assuming that the lines of the program are enumerated in increasing order, this corresponds to the lim inf rule $S(t) = \liminf_{s\to t} S(s)$. The natural head location $H(t)$ is then determined as the inferior limit of all those head locations when the program is at the start of the "main loop".

A computation of the α-machine may be visualized by a "space–time" diagram like:



				S	p	a	c	e		α						
		0	1	2	3	4	5	6	7	…	…	ω	…	$\theta<\alpha$	…	…
	0	**1**	1	0	1	0	0	1	1	…	…	1	…	0	0	…
	1	0	**1**	0	1	0	0	1	1			1				
T	2	0	0	**0**	1	0	0	1	1			1				
i	3	0	0	0	**1**	0	0	1	1			1				
m	4	0	0	0	0	**0**	0	1	1			1				
e	⋮															
	n	1	1	1	1	**0**	1	0	1			1				
α	n+1	1	1	1	**1**	1	1	0	1			1				
	⋮	⋮	⋮	⋮	⋮	⋮										
	ω	0	0	1	0	0	0	1	1	…	…	**1**				
	$\omega+1$	**0**	0	1	0	0	0	1	1			0				
	⋮		**⋱**													
	$\theta-1$	1	0	0	1	1	1	**1**	0	…	…	…	…	0	…	…
	$\theta<\alpha$	S	T	O	P											

A computation of an α-machine, head positions are indicated by shading.

The computation

$$(T(t), H(t), S(t))_{t<\theta}$$

defined above is called the *α-computation by P with input $T(0)$*. If the α-computation stops at some $\theta < \alpha$ then $\theta = t_0 + 1$ is a successor ordinal and $T(t_0)$ is the final tape content. In this case we say that *P computes $T(t_0)$* from $T(0)$ and write

$P : T(0) \mapsto T(t_0)$. Note that at all times t the head position $H(t)$ is $\leq t$ and that $T(t) = (T(t) \upharpoonright t) \cup (T(0) \upharpoonright [t, \alpha))$. So the α-computation can be described by $T(0)$ and the sequence

$$(T(t) \upharpoonright t, H(t), S(t))_{t<\theta}.$$

We can define various notions of *computability* from this machine model. Information is entered and output on the one working tape of the machine. Instead of introducing several machine tapes we separate various kinds of information on the tape by dividing it up into four "subtapes", using ordinal arithmetic modulo 4: ordinals $\equiv 0 \pmod 4$ are used to code the "input" $X \subseteq \alpha$, ordinals $\equiv 1 \pmod 4$ code the "output" $Y \subseteq \alpha$, ordinals $\equiv 2 \pmod 4$ code extra "parameters" $p \subseteq \alpha$, and ordinals $\equiv 3 \pmod 4$ may contain an "oracle" $O \subseteq \alpha$. Appropriate codings and decodings are given by the characteristic function $X|Y|p|O : \alpha \to 2$, where for $\xi < \alpha$ and $i < 4$

$$4 \cdot \xi + i \mapsto 1 \text{ iff } (i = 0 \wedge \xi \in X) \vee (i = 1 \wedge \xi \in Y) \vee (i = 2 \wedge \xi \in p) \vee (i = 3 \wedge \xi \in O).$$

We can use an α-machine to (effectively) enumerate all the pairs (P, p) where P is a Turing program and $p \subseteq \alpha$ a finite set of ordinal parameters. Fix such an enumeration.

Definition 1. Let $\epsilon < \alpha$ and denote by $\{\epsilon\}$ the ϵth pair (P, p), consisting of a program P and parameters p. Let $\delta \in \alpha$, $B \subseteq \alpha$ and $\sigma < \alpha$. Consider $(T(t), H(t), S(t))_{t<\theta}$, the α-computation by P on input $T(0) = \{\delta\}|\emptyset|p|B$.

- $\{\epsilon\}^B_\sigma(\delta)\downarrow$ means that the computation halts in at most σ-many stages, i.e. $\theta \leq \sigma$.
- $\{\epsilon\}^B(\delta)\downarrow$ means that the computation halts, i.e. $\theta < \alpha$.
- $\{\epsilon\}^B(\delta)\uparrow$ means that the computation *diverges*, i.e. $\theta = \alpha$.
- We write $\{\epsilon\}^B_\sigma(\delta)\downarrow = \gamma$ iff $\{\epsilon\}^B_\sigma(\delta)\downarrow$ and $P : \{\delta\}|\emptyset|p|B \mapsto X'|\{\gamma\}|p|B$ for some $X' \subseteq \alpha$ and furthermore for all $t < \sigma \cap \theta$ $T(t)$ is of the form $X_t|Y_t|p|B$ for some $X_t, Y_t \subseteq \alpha$.

If $B = \emptyset$ we write $\{\epsilon\}$ instead of $\{\epsilon\}^B$.

Definition 2. A partial function $F : \alpha \rightharpoonup \alpha$ is *α-computable in (the oracle) $B \subseteq \alpha$* iff there is an $\epsilon < \alpha$ such that for all $\delta < \alpha$:

- $\{\epsilon\}^B(\delta)\downarrow$ iff $\delta \in \mathrm{dom}(F)$
- $\{\epsilon\}^B(\delta)\downarrow = F(\delta)$ for every $\delta \in \mathrm{dom}(F)$.

In that case we say that $\{\epsilon\}^B$ computes F and write $\{\epsilon\}^B = F$.

A set $A \subseteq \alpha$ is *α-computable in (the oracle) B* iff its characteristic function $\chi_A : \alpha \to 2$ is α-computable in B. A set $A \subseteq \alpha$ is *α-computably enumerable in (the oracle) B* iff $A = \mathrm{dom}(F)$ for some partial function $F : \alpha \rightharpoonup 2$ which is α-computable in B. In case $B = \emptyset$ we simply write *α-computable* and *α-computably enumerable*.

We also write *ordinal machine*, *ordinal computable*, and *ordinal computably enumerable* instead of ∞-machine, ∞-computable, and ∞-computably enumerable, resp.

These are the basic notions of *α-computability theory*, defined in close analogy to the notions of classical computability or recursion theory. To study the relation of α-computability theory to α-recursion theory we link α-computability to constructibility theory. Since an α-computation is defined by very simple recursion rules it can be carried out within the levels $L_\delta[\cdot]$ of appropriate relativized constructible hierarchies.

Lemma 3. *Let P be a program and assume that the initial tape content is the characteristic function of a set $D \subseteq \alpha$: $T(0) = \chi_D$. Let $(T(t), H(t), S(t))_{t<\theta}$ be the α-computation by P with input $T(0)$. Then:*

(a) *If δ is a limit ordinal then $\forall \nu < \delta\ (T(t) \upharpoonright t, H(t), S(t))_{t\in\theta\cap\nu} \in L_\delta[D]$.*
(b) *If δ is a limit ordinal then $(T(t) \upharpoonright t, H(t), S(t))_{t\in\theta\cap\delta}$ is uniformly $\Delta_1(L_\delta[D])$.*
(c) *If $A \subseteq \alpha$ is α-computably enumerable in the oracle B then it is $\mathbf{\Sigma}_1(L_\alpha[B])$.*
(d) *If $A \subseteq \alpha$ is α-computable in B then it is $\mathbf{\Delta}_1(L_\alpha[B])$.*
(e) *If $A \subseteq \alpha$ is α-computably enumerable then it is $\mathbf{\Sigma}_1(L_\alpha)$.*
(f) *If $A \subseteq \alpha$ is α-computable then it is $\mathbf{\Delta}_1(L_\alpha)$.*

Proof. We prove (a) and (b) by simultaneous induction on δ. (a) holds readily for $\delta = \omega$ since $L_\omega[D] = H_\omega$ is the set of all hereditarily finite sets.

We assume the (a) holds at δ and show that (b) holds at δ. As remarked above,

$$(T(t) \upharpoonright t, H(t), S(t))_{t\in\theta\cap\delta}$$

is basically the α-computation by P with input $T(0)$, restricted to $\theta \cap \delta$. It is defined by the recursive computation rules by the program P with input $T(0)$. The recursive rules can be defined by Σ_0-formulas, and so by the recursion theorem for ordinal recursion $(T(t) \upharpoonright t, H(t), S(t))_{t\in\theta\cap\delta}$ is Δ_1-definable in the set-theoretic universe. The unbounded quantifiers in the Δ_1-representation range over initial segments of the recursive functions. By (a), these initial segments are elements of $L_\delta[D]$ and so $(T(t) \upharpoonright t, H(t), S(t))_{t\in\theta\cap\delta}$ is $\Delta_1(L_\delta[D])$.

Now assume that δ is a limit ordinal such that (a) and (b) hold for all limit ordinals $\delta' < \delta$. We show that (a) holds at δ. This is obvious if δ is a limit of limit ordinals. Assume now that $\delta = \delta' + \omega$, where δ' is a limit ordinal. If $\theta < \delta'$, (a) holds trivially at

δ. So assume $\theta \geq \delta'$. Then $(T(t) \upharpoonright t, H(t), S(t))_{t<\delta'}$ is uniformly $\Delta_1(L_{\delta'}[D])$. The limit configuration $(T(\delta') \upharpoonright t, H(\delta'), S(\delta'))$ is definable by simple lim inf rules from $(T(t) \upharpoonright t, H(t), S(t))_{t<\delta'}$. So it is also definable over $L_{\delta'}[D]$ and

$$(T(t) \upharpoonright t, H(t), S(t))_{t\leq\delta'} \in L_{\delta'+1}[D].$$

For $\nu \in [\delta', \delta)$, the sequences $(T(t) \upharpoonright t, H(t), S(t))_{t\in\theta\cap\nu}$ are "finite variations" of $(T(t) \upharpoonright t, H(t), S(t))_{t\leq\delta'}$ and hence $(T(t) \upharpoonright t, H(t), S(t))_{t\in\theta\cap\nu} \in L_{\delta'+\omega}[D]$, as required by (b).

Now (b) implies (c), (c) implies (d), (c) implies (e), and (d) implies (f). □

To prove the converse of (c) and (d), we represent the constructible hierarchy $L_\delta[D]$, for $D \subseteq$ Ord by an ordinal program. According to [5], basic functions like the Gödel-pairing function are ordinal computable. This allows to code sequences of ordinals and formulas as ordinals; elementary operations on sequences can be assumed to be ordinal computable.

To make $L_\alpha[D]$ accessible to an α-Turing machine introduce a language with symbols (,), {, }, |, $\in$, $=$, $\wedge$, $\neg$, $\forall$, $\exists$, $\dot{B}$ and variables $v_0, v_1, \ldots$. Define (*bounded*) *formulas* and (*bounded*) *terms* by a common recursion on the lengths of words formed from these symbols:

- the variables $v_0, v_1, \ldots$ are terms;
- if s and t are terms then $\dot{B}(s)$, $s = t$ and $s \in t$ are formulas;
- if φ and ψ are formulas then $\neg\varphi$, $(\varphi \wedge \psi)$, $\forall v_i \in v_j\, \varphi$ and $\exists v_i \in v_j\, \varphi$ are formulas;
- if φ is a formula then $\{v_i \in v_j | \varphi\}$ is a term.

For technical reasons we use *tidy* terms and formulas in which

- no bound variable occurs free,
- every free variable occurs exactly once.

An *assignment* is a finite sequence $a : k \to V$; $a(i)$ will be the *interpretation* of the variable v_i. We write $t[a]$ and $\varphi[a]$ for the values of t and φ under the assignment a. Concerning the constructible hierarchy $L_\delta[D]$, it can be shown by induction on δ that every element of $L_\delta[D]$ is the interpretation $t[(L_{\gamma_0}[D], \ldots, L_{\gamma_{k-1}}[D])]$ of some *tidy* term t with an assignment $(L_{\gamma_0}[D], \ldots, L_{\gamma_{k-1}}[D])$ where $\gamma_0, \ldots, \gamma_{k-1} < \delta$.

We define the *(bounded) truth function* $W_D : A \to 2$ for $L[D]$ on the class

$$A = \{(a, \varphi) | a \in \mathrm{Ord}^{<\omega}, \varphi \text{ is a tidy bounded formula}\};$$

$$W_D((\gamma_0, \ldots, \gamma_{k-1}), \varphi) = 1 \text{ iff } \varphi[(L_{\gamma_0}[D], \ldots, L_{\gamma_{k-1}}[D])].$$

Relativizing the main technical result of [5] to the oracle D yields:

Lemma 4. *The bounded truth function W_D for $L[D]$ is ordinal computable in the oracle D by some Turing program P_{truth}.*

A close inspection of the program P_{truth} shows that the computation takes place in *exponential time*, i.e., there is an exponential expression $p(\xi)$ such that for $((\gamma_0, \ldots, \gamma_{k-1}), \varphi) \in A$ the computation of the truth value $W((\gamma_0, \ldots, \gamma_{k-1}), \varphi)$ stops before stage $p(\max(\gamma_0, \ldots, \gamma_{k-1}))$, where the exponential expression is evaluated in ordinal arithmetic. This proves:

Lemma 5. *If the ordinal $\alpha > 0$ is closed with respect to ordinal exponentiation then the bounded truth function $W_D \upharpoonright (A \cap (\alpha^{<\omega} \times V))$ is α-computable in the oracle $D \cap \alpha$.*

Lemma 6. *Let $\alpha > 0$ be closed with respect to ordinal exponentiation and let $A \subseteq \alpha$ be $\mathbf{\Sigma}_1(L_\alpha[B])$. Then A is α-computably enumerable in the oracle B. If $A \subseteq \alpha$ is $\mathbf{\Delta}_1(L_\alpha[B])$ then A is α-computable in the oracle B.*

Proof. Consider a $\mathbf{\Sigma}_1(L_\alpha[B])$-definition of $A \subseteq \alpha$:

$$\xi \in A \leftrightarrow \exists y \in L_\alpha[B]\ L_\alpha[B] \models \varphi[\xi, y, \vec{a}]$$

where φ is a bounded formulas. This is equivalent to

$$\xi \in A \leftrightarrow \exists \beta < \alpha\ L_\beta[B] \models \exists y \varphi[\xi, y, \vec{a}]$$

and

$$\xi \in A \leftrightarrow \exists \beta < \alpha\ \ W_B((\xi, \beta, \vec{a}), \varphi^*) = 1$$

where φ^* is an appropriate tidy formula.

Now $\xi \in A$ is α-computably enumerable in B by the following "search procedure": for $\gamma < \alpha$ and for $\beta < \gamma$ let the program P_{truth} run on input $((\xi, \beta, \vec{a}), \varphi^*)$ for γ steps. If the truth program stops with output 1, then stop, otherwise continue.

For the second part, let $A \subseteq \alpha$ be $\mathbf{\Delta}_1(L_\alpha[B])$. Then A and $\alpha \setminus A$ are α-computably enumerable in B, and hence A is α-computable in B. □

The results so far yield the following characterizations:

Theorem 7. *Let the ordinal α be closed under ordinal exponentiation and $A \subseteq \alpha$. Then*

(a) *A is α-computable in B iff A is $\mathbf{\Delta}_1(L_\alpha[B])$.*
(b) *A is α-computably enumerable in B iff A is $\mathbf{\Sigma}_1(L_\alpha[B])$.*

3. Admissible recursion theory

Admissible ordinals were defined by Kripke [7] in order to generalize standard recursion theory to ordinals. One of the many equivalent definitions is:

Definition 8. An ordinal $\alpha \geq \omega$ is *admissible* iff there is no total $\mathbf{\Sigma}_1(L_\alpha)$-definable function f that maps an ordinal $\beta < \alpha$ cofinally into α.

Note that every admissible ordinal is closed under ordinal exponentiation. By Theorem 7, we characterize admissibility in terms of ordinal computability without any reference to the constructible hierarchy.

Theorem 9. *An ordinal α closed under ordinal exponentiation is admissible iff there is no α-computable function g that maps some $\beta < \alpha$ cofinally into α.*

In case of an admissible ordinal α, the absence of computable functions cofinal in α enables us to make free use of nested loops of lengths $< \alpha$ and finitely many working tapes with independent heads when describing algorithms for α-machines. Recall the fundamental definitions of admissible recursion theory (see [11, p. 154–155]):

Definition 10. Let α be admissible. Then

(a) x is *α-finite* iff $x \in L_\alpha$.
(b) $A \subseteq \alpha$ is *α-recursive* iff A is $\mathbf{\Delta}_1(L_\alpha)$.
(c) $A \subseteq \alpha$ is *α-recursively enumerable (α-r.e.)* iff A is $\mathbf{\Sigma}_1(L_\alpha)$.

Theorem 7 immediately gives the equivalences:

Theorem 11. *Let α be admissible. Then*

(a) *$A \subseteq \alpha$ is α-recursive iff A is α-computable.*
(b) *$A \subseteq \alpha$ is α-r.e. iff A is α-computably enumerable.*

The notion of α-finiteness can be characterized by means of α-computability in the following way:

Theorem 12. *Let α be admissible. For $A \subseteq \alpha$ it is equivalent:*

(a) *A is α-finite ,*
(b) *A is α-computable and bounded below α,*
(c) *A is β-computable for some $\beta < \alpha$.*

Proof. (a) ⇒ (b) since α-finite implies α-computable and α is a limit ordinal.
(b) ⇒ (c). Let $A \subseteq \gamma < \alpha$ and let $\{\epsilon\} = \chi_A$. Since α is admissible there exists an upper bound $\delta < \alpha$ on the length of the computations $\{\epsilon\}(\xi)$ for $\xi < \gamma$. So A is $\max\{\delta, \gamma\}$-computable.
(c) ⇒ (a). We may assume that β is a limit ordinal. By Lemma 3 A has a Δ_1-definition over L_β. So $A \in L_{\beta+1} \subseteq L_\alpha$. □

Identifying a function f with its graph $\{\langle \zeta, \eta \rangle \mid f(\zeta) = \eta\}$ (where $\langle \cdot, \cdot \rangle$ is the Gödel-pairing function), we have a natural notion of α-finite functions on ordinals. Also, the cardinals within L_α can now be characterized by α-machines.

Theorem 13. *Let α be admissible and $\beta < \alpha$. Then*

(a) *$L_\alpha \models$ 'β is a cardinal' iff there is no $\delta < \beta$ and no α-finite $g : \delta \xrightarrow{surj} \beta$. We call β an α-*cardinal.
(b) *$L_\alpha \models$ 'β is a regular cardinal' iff β is an α-cardinal and there is no $\gamma < \beta$ with an α-finite $h : \gamma \xrightarrow{cof} \beta$. In this case we say β is an α-*regular cardinal.

Definition 14. For α-finite $A \subseteq \alpha$ we denote by $|A|_\alpha$ the *α-cardinality* of A, i.e. the least ordinal $\delta < \alpha$ with an α-finite $g : \delta \xrightarrow{surj} A$.

Admissible recursion theory uses methods from Jensen's fine structure of the constructible hierarchy [4] as a partial substitute for the strong closure properties of ω. The central notion of fine structure theory is given by the *projectum*. The following definition taken from [2] is equivalent to the original one.

Definition 15. For every ordinal $\alpha \geq \omega$ define its *projectum*

$$\alpha^* = \min\{\rho \leq \alpha \mid \exists g \; (g \; \mathbf{\Sigma}_1(L_\alpha) \wedge g : L_\rho \xrightarrow{surj} L_\alpha)\}.$$

Theorem 7 yields a characterization without reference to the constructible hierarchy.

Theorem 16. *Let α be admissible. Then α^* is the smallest ordinal such that there is an α-computable injection from α into α^*.*

Lemma 17.

$$\alpha^* = \min\{\rho \leq \alpha \mid \exists \tilde{g}\ (\tilde{g}\ \mathbf{\Sigma}_1(L_\alpha) \wedge \tilde{g} : \rho \xrightarrow{surj} \alpha)\}.$$

Proof of Lemma 17. Let g be $\mathbf{\Sigma}_1(L_\alpha)$ with $g : L_\rho \xrightarrow{surj} L_\alpha$. There exists a $\mathbf{\Delta}_1(L_\alpha)$ $h : \alpha \overset{bij}{\to} L_\alpha$ [11, Proposition 1.8, p. 156]. Similarly, we can obtain a $\mathbf{\Delta}_1(L_\alpha)$-definable $\tilde{h} : \rho \overset{bij}{\to} L_\rho$. Define $\tilde{g} = h^{-1} \circ g \circ \tilde{h}$. This is $\mathbf{\Sigma}_1(L_\alpha)$ and $\tilde{g} : \rho \xrightarrow{surj} \alpha$. Conversely, let $\tilde{g}$ be given as $\mathbf{\Sigma}_1(L_\alpha)$ and with $\tilde{g} : \rho \xrightarrow{surj} \alpha$. Then $g = h \circ \tilde{g} \circ \tilde{h}^{-1}$ is $\mathbf{\Sigma}_1(L_\alpha)$ and $g : L_\rho \xrightarrow{surj} L_\alpha$. □

Proof of Theorem 16. Since α is admissible assume that we have a partial α-computable $g : \rho \xrightarrow{surj} \alpha$. We define the α-computable $f : \alpha \overset{inj}{\to} \rho$ as follows: Given input ξ, we check in stage σ the first σ-many steps of the computations of $g(\iota)$ for $\iota < \sigma$. At some least stage we will find a least η with $g(\eta) = \xi$ and set $f(\xi) = \eta$. If on the other hand we assume a $f : \alpha \overset{inj}{\to} \beta$, it suffices to define a partial α-computable $g : \beta \xrightarrow{surj} \alpha$: Set $g(\xi)$ to the η with $f(\eta) = \xi$ and undefined else. □

Admissible recursion theory studies subsets of α with respect to certain reducibility relations. We define three reducibility notions. The first two are standard in α-recursion theory (see [11, p. 162]), the third is the natural notion arising from α-computability. Fix an admissible ordinal α for the rest of this section.

Definition 18. (a) A is *weakly α-recursive in B*, $A \leq_{w\alpha} B$, iff there exists an α-recursively enumerable set $R \subseteq L_\alpha$ such that for all $\gamma < \alpha$

$$\gamma \in A \text{ iff } \exists H \subseteq B \exists J \subseteq \alpha \setminus B : \quad (H, J, \gamma, 1) \in R$$

and

$$\gamma \notin A \text{ iff } \exists H \subseteq B \exists J \subseteq \alpha \setminus B : \quad (H, J, \gamma, 0) \in R.$$

(b) A is *α-recursive in B*, $A \leq_\alpha B$, iff there exist α-r.e. sets $R_0, R_1 \subseteq L_\alpha$ such that for all $K \in L_\alpha$

$$K \subseteq A \text{ iff } \exists H \subseteq B \exists J \subseteq \alpha \setminus B : \quad (H, J, K) \in R_0$$

and

$$K \subseteq \alpha \setminus A \text{ iff } \exists H \subseteq B \exists J \subseteq \alpha \setminus B : \quad (H, J, K) \in R_1.$$

Definition 19. For subsets $A, B \subseteq \alpha$ define $A \preceq_\alpha B$, A is *α-computably reducible* to B, iff A is α-computable in the oracle B.

By Theorem 7 this can be reformulated as:

Theorem 20. *For $A, B \subseteq \alpha$: $A \preceq_\alpha B$ iff $A \in \mathbf{\Delta}_1(L_\alpha[B])$.*

It is easy to see that $A \leq_\alpha B$ implies $A \leq_{w\alpha} B$. The relation $\leq_\alpha$ is transitive, whereas $\leq_{w\alpha}$ and $\preceq_\alpha$ may fail to be transitive (see [3]). Inspection of the definition of $\leq_{w\alpha}$ shows:

Theorem 21. *If A is weakly α-recursive in B then A is α-computable in B.*

Thus we have the following inclusions

$$\leq_\alpha \subseteq \leq_{w\alpha} \subseteq \preceq_\alpha .$$

The inclusions can in general not be reversed.

4. The Sacks–Simpson theorem with machines

The first major result in α-recursion theory was Sacks' and Simpson's proof of a positive answer to Post's problem for α-recursion:

Theorem 22 (*Sacks–Simpson 1972*). *Let α be an admissible ordinal. Then there are two α-r.e. sets $A \subseteq \alpha$ and $B \subseteq \alpha$ such that $A \not\leq_{w\alpha} B$ and $B \not\leq_{w\alpha} A$.*

Post's problem is usually seen as a test case for recursion theories. So we are interested whether or not some priority argument can be implemented on our α-machines. More precisely, can we utilize the lim inf-rule in a way that the algorithm behaves correctly at limit times, preserving all the necessary information? We will see that it is instrumental to store information in different data structures (see (1) and (2) below).

We base our considerations on the presentations of the Sacks–Simpson theorem in [11,12]. There, the proof is divided in two cases, depending on the closure properties of the admissible ordinal α. Each case is proved by describing an algorithm that simultaneously enumerates the two sets A and B as required.

The first case, $\alpha^* < \alpha$, uses a generalization of the classical Friedberg–Muchnik algorithm, but with requirements indexed by α^*. In the correctness proof, an analysis of the α-cardinality structure ensures that all injury sets are α-finite. Instead of employing the complicated method of tame Σ_2-maps as in [11,12], an adaptation of the method described by Shore in [13] can be used to handle the case that $\alpha^* = \alpha$, as briefly suggested in the non-trivial Exercise VIII.1.5 in [11]: The above algorithm is modified by adding an additional set of negative requirements which keep the injury sets literally finite. The closure of α under partial Σ_1-maps ensures the correctness despite the heavily restricted injury behavior.

The techniques to implement either of these algorithms on α-machines are the same, therefore we choose to restrict our presentation to the algorithm for the case $\alpha^* < \alpha$.

From now on we assume $\alpha^* < \alpha$.

By Theorem 21, the Sacks–Simpson theorem can be proved by giving an algorithm that computably enumerates two sets A and B, for which the goals $A \not\leq_\alpha B$ and $B \not\leq_\alpha A$ hold. Consider the following requirements:

$$\begin{aligned} \mathrm{req}_{2\epsilon} &: \text{If } \epsilon \in \mathrm{ran}(f) \text{ then } \{f^{-1}(\epsilon)\}^B \neq \chi_A \\ \mathrm{req}_{2\epsilon+1} &: \text{If } \epsilon \in \mathrm{ran}(f) \text{ then } \{f^{-1}(\epsilon)\}^A \neq \chi_B \end{aligned}$$

with a fixed map $f : \alpha \overset{inj}{\rightarrow} \alpha^*$ according to Theorem 16.

Along with the sets A and B the algorithm will give a *witness* for every $\mathrm{req}_{2\epsilon}$ with $\epsilon \in \mathrm{ran}(f)$, i.e., an ordinal $w_{2\epsilon}$ for which

$$\{f^{-1}(\epsilon)\}^B(w_{2\epsilon}) \downarrow = 0 \text{ iff } w_{2\epsilon} \in A$$

and likewise a witness $w_{2\epsilon+1}$ for $\mathrm{req}_{2\epsilon+1}$.

At any given stage $\sigma < \alpha$ of the algorithm we have the following data for every requirement $\mathrm{req}_{2\epsilon}$ (making use of the α-computable Gödel-pairing function $\langle \cdot, \cdot \rangle$ for coding):

(1) $w^\sigma_{2\epsilon}$ is the candidate for the witness for $\mathrm{req}_{2\epsilon}$ at stage σ. Those candidate witnesses only *increase* over time, so we can represent them on one tape in a way that at any given time σ the tape contains a 1 in cell $\langle 2\epsilon, \gamma\rangle$ iff $\gamma < w^\sigma_{2\epsilon}$. Instead of the single ordinal $w^\sigma_{2\epsilon}$, the complete initial segment of ordinals up to $w^\sigma_{2\epsilon}$ is coded onto the tape.
(2) $\mathrm{used}^\sigma_{2\epsilon}$ is non-zero iff $w^\sigma_{2\epsilon} \in A$ and is set to the supremum over the cell indices used in a the computation responsible for putting $w^\sigma_{2\epsilon}$ into A or B. Those are stored on one tape whose $\langle 2\epsilon, \beta\rangle$th cell contains a 1 iff $\mathrm{used}^\sigma_{2\epsilon} = \beta$. We want $\mathrm{used}^\sigma_{2\epsilon} = 0$ to code the information that $w^\sigma_{2\epsilon} \notin A$. The variable $\mathrm{used}^\sigma_{2\epsilon}$ is represented as a single ordinal bit on the tape.

All of these are analogously defined for $\mathrm{req}_{2\epsilon+1}$ with the roles of A and B interchanged. A and B are constructed as their characteristic functions on one tape each, A^σ resp. B^σ denote the part constructed at the beginning of stage σ. The candidate witnesses w^σ_δ for req_δ are chosen from the class $Z_\delta = \{\langle \delta, \xi\rangle \mid \xi \in \mathrm{Ord}\}$ so w^σ_δ can only enter A or B as a witness for req_δ.

In what follows, whenever a statement is made about $\mathrm{req}_{2\epsilon}$, its dual for $\mathrm{req}_{2\epsilon+1}$ is also assumed yet omitted for simplicity. We say that $\mathrm{req}_{2\epsilon}$ is *not currently witnessed* at stage σ if $w^\sigma_{2\epsilon} \notin A^\sigma$. $\mathrm{req}_{2\epsilon}$ *acts* at stage σ when $w^\sigma_{2\epsilon}$ is added to A at stage σ. As long as $w^\sigma_{2\epsilon} \in A^\sigma$ we call $\mathrm{req}_{2\epsilon}$ *currently witnessed*. $\mathrm{req}_{2\epsilon}$ is *injured* by req_δ in stage σ if $w^\sigma_{2\epsilon} \in A^\sigma$ but $w^{\sigma+1}_{2\epsilon} \notin A^{\sigma+1}$ and req_δ is the requirement acting in stage σ.

The following Friedberg–Muchnik-type algorithm will check each requirement unboundedly often, but every requirement will act only α-finitely many times [12].

Algorithm 1. The algorithm goes through α-many stages σ and in each stage it might put an element into A or B to fulfill one requirement (if so, we say that the respective requirement acts).

$\sigma = 0$: Initialize $w^0_0 = \langle 0, 0\rangle$ and $\mathrm{used}^0_0 = 0$.

$\sigma \mapsto \sigma + 1$: Let $\sigma = \mu \cdot \alpha^* + \nu$ with $\nu < \alpha^*$. If w^σ_ν does not have a value, initialize $w^\sigma_\nu = \min\{\xi \in Z_\nu \mid \xi \notin A^\sigma \cup B^\sigma \wedge \xi > \sup\{\mathrm{used}^\sigma_\gamma \mid \gamma < \nu\}\}$ and $\mathrm{used}^\sigma_\nu = 0$. [This initialization happens exactly at stages $< \alpha^*$.]

Let WLOG $\nu = 2\epsilon$.

If $w^\sigma_{2\epsilon} \in A^\sigma$ then $\mathrm{req}_{2\epsilon}$ is currently witnessed and no action is taken, i.e., set $A^{\sigma+1} = A^\sigma$, $B^{\sigma+1} = B^\sigma$, $w^{\sigma+1}_\delta = w^\sigma_\delta$ and $\mathrm{used}^{\sigma+1}_\delta = \mathrm{used}^\sigma_\delta$ for all $\delta < \alpha^*$.

If $w^\sigma_{2\epsilon} \notin A$ then check whether $\{f^{-1}(\epsilon)\}^{B^\sigma}_\sigma(w^\sigma_{2\epsilon}) \downarrow = 0$.

If this is not true then again no action is taken.

If $\{f^{-1}(\epsilon)\}^{B^\sigma}_\sigma(w^\sigma_{2\epsilon}) \downarrow = 0$ then $\mathrm{req}_{2\epsilon}$ acts: set $A^{\sigma+1} = A^\sigma \cup \{w^\sigma_{2\epsilon}\}$, $B^{\sigma+1} = B^\sigma$, $w^{\sigma+1}_{2\epsilon} = w^\sigma_{2\epsilon}$. Let u be the supremum of the indices of cells used in the computation of $\{f^{-1}(\epsilon)\}^{B^\sigma}_\sigma(w^\sigma_{2\epsilon}) \downarrow = 0$ and set $\mathrm{used}^{\sigma+1}_{2\epsilon} = u$. For $\delta < 2\epsilon$ keep $w^{\sigma+1}_\delta = w^\sigma_\delta$, $\mathrm{used}^{\sigma+1}_\delta = \mathrm{used}^\sigma_\delta$. For $\alpha^* > \delta > 2\epsilon$ and $\delta \leq \sigma$: $\mathrm{used}^{\sigma+1}_\delta = 0$, $w^{\sigma+1}_\delta = \min\{\xi \in Z_\delta \mid \xi \notin A^{\sigma+1} \cup B^{\sigma+1} \wedge \xi > w^\sigma_\delta \wedge \xi > u\}$.

Note that the description of the algorithm does not contain a limit step since the machine configuration at limit times is completely determined by the lim inf-rules.

We need to check whether the variables contain their intended values also at limit stages:

Lemma 23. *At the beginning of every stage σ the following conditions hold for every $\delta < \alpha^*$ resp. $2\epsilon < \alpha^*$:*

(a) $w^\sigma_\delta > \sup\{\mathrm{used}^\sigma_\gamma \mid \gamma < \delta\}$,
(b) *If $w^\sigma_{2\epsilon} \in A$ then $\{f^{-1}(\epsilon)\}^{B^\sigma}_\sigma(w^\sigma_{2\epsilon}) \downarrow = 0$ and $\mathrm{used}^\sigma_{2\epsilon}$ is the supremum over the indices of the cells used in this computation.*

Proof. If σ is a successor ordinal, the lemma holds by the definition of w and used. If σ is a limit consider a requirement $\mathrm{req}_{2\epsilon}$. If $\mathrm{req}_{2\epsilon}$ acted at a stage $\tau < \sigma$ and no requirement req_{δ} with $\delta < 2\epsilon$ acted at a stage $\rho, \tau < \rho < \sigma$, then $w^{\tau}_{2\epsilon} = w^{\sigma}_{2\epsilon}$ and (a) holds. In fact $\{f^{-1}(\epsilon)\}^{B^{\sigma}}_{\sigma}(w^{\sigma}_{2\epsilon}) \downarrow= 0$ is the same computation as $\{f^{-1}(\epsilon)\}^{B^{\tau}}_{\tau}(w^{\tau}_{2\epsilon}) \downarrow= 0$ since the elements in $B^{\sigma} \setminus B^{\tau}$ are all larger than the supremum of cell indices used in $\{f^{-1}(\epsilon)\}^{B^{\tau}}_{\tau}(w^{\tau}_{2\epsilon}) \downarrow= 0$. Since $\mathrm{used}^{\tau}_{2\epsilon} = \mathrm{used}^{\sigma}_{2\epsilon}$ also (b) holds.

If, on the other hand, $\mathrm{req}_{2\epsilon}$ was injured unboundedly often in σ, then, by the lim inf-rule, $\mathrm{used}^{\sigma}_{2\epsilon} = 0$ and $w_{2\epsilon}$ has been redefined unboundedly often in σ. Since, unlike $\mathrm{used}_{2\epsilon}$, $w_{2\epsilon}$ is represented as an initial segment of '1's on the tape, it does not default to zero but takes its intended value at stage σ. By admissibility $w^{\sigma}_{2\epsilon}$ is well defined, i.e. $< \alpha$. So (a) holds inductively. Since $w^{\sigma}_{2\epsilon} \notin A$, (b) is trivially true in this case. □

Seeing that the data structures behave well at limit times, it is clear that this algorithm is similar to the ones from [12] and [11] and therefore correctly solves Post's problem for α-recursion theory in case that $\alpha^{*} < \alpha$.

References

[1] Chi Tat Chong, Sy D. Friedman, Ordinal recursion theory, in: Edward R. Griffor (Ed.), Handbook of Computability Theory, North Holland, 1999, pp. 277–299.
[2] Keith Devlin, Constructibility, Springer, Berlin, 1984.
[3] Graham C. Driscoll Jr., Metarecursively enumerable sets and their metadegrees, The Journal of Symbolic Logic 33 (1968) 389–411.
[4] Ronald B. Jensen, The fine structure of the constructible hierarchy, Annals of Mathematical Logic 4 (1972) 229–308.
[5] Peter Koepke, Turing computations on ordinals, The Bulletin of Symbolic Logic 11 (2005) 377–397.
[6] Georg Kreisel, Gerald E. Sacks, Metarecursive sets, The Journal of Symbolic Logic 30 (1965) 318–338.
[7] Saul Kripke, Transfinite recursion on admissible ordinals I, II (Abstracts), The Journal of Symbolic Logic 29 (1964) 161–162.
[8] Azriel Levy, Transfinite computability, Notices of the American Mathematical Society 10 (1963) 286.
[9] Moshé Machover, The theory of transfinite recursion, Bulletin of the American Mathematical Society 67 (1961) 575–578.
[10] Richard Platek, Foundations of recursion theory, Ph.D. Thesis, Stanford University, 1965.
[11] Gerald E. Sacks, Higher Recursion Theory, Springer, Berlin, Heidelberg, 1990.
[12] Gerald E. Sacks, Stephen G. Simpson, The α-finite injury method, Annals of Mathematical Logic 4 (1972) 343–367.
[13] Richard A. Shore, Some more minimal pairs of α-recursively enumerable degrees, Zeitschrift für Mathematische Logik und Grundlagen der Mathematik 24 (1978) 409–418.
[14] Gaisi Takeuti, On the recursive functions of ordinal numbers, Journal of the Mathematical Society of Japan 12 (1960) 119–128.