Ordinal Computability

BY PETER KOEPKE

University of Bonn
Mathematisches Institut, Beringstraße 1, D 53115 Bonn, Germany

Email: koepke@math.uni-bonn.de

February 7, 2009

Abstract

Ordinal computability uses ordinals instead of natural numbers in abstract machines like register or TURING machines. We give an overview of the computational strengths of α-β-machines, where α and β bound the time axis and the space axis of some machine model. The spectrum ranges from classical TURING computability to ∞-∞-computability which corresponds to GÖDEL's model of constructible sets. To illustrate some typical techniques we prove a new result on Infinite Time Register Machines (= ∞-ω-register machines) which were introduced in [5]: a real number $x \in {}^{\omega}2$ is computable by an Infinite Time Register Machine iff it is TURING computable from some finitely iterated hyperjump $0^{(n)}$.

KEYWORDS: Ordinal machines, Infinite Time Register Machines

1 Ordinal computations [1]

Standard computability theory is fundamentally based on the set $\mathbb{N} = \{0, 1, 2, ...\}$ of natural numbers: steps of a computation are clocked by natural numbers ("time"), cells of a TURING tape are indexed by natural numbers ("space"), and read and write registers contain natural numbers ("space"). The canonical wellorder $0 < 1 < 2 < ...$ of natural numbers allows recursive algorithms, the arithmetic properties of the structure $(\omega, +, \cdot, 0, 1)$ allow to code various information. A standard TURING computation can be represented by a space-time function or diagram with natural numbers axes.

	⋮										
	⋮				⋮		⋮			⋰	
↑	$n+1$	0	0	0	0	0	⋯	0	0	⋯	
	n	0	0	0	0	0	⋯	0	1	⋯	
S	⋮	⋮	⋮	⋮	⋮	⋮	⋰	⋮	⋮		
P	4	1	1	1	1	1	⋯	1	1		
A	3	1	1	1	1	1	⋯	1	1		
C	2	0	0	0	1	1	⋯	1	1		
E	1	0	0	1	1	0	⋯	0	0		
	0	1	0	0	0	0	⋯	0	0		
		0	1	2	3	4	⋯	m	$m+1$	⋯	⋯
				T	I	M	E		→		

1. The author wants to thank JOEL HAMKINS and PHILIP WELCH for a very inspiring discussion at the EMU 2008 workshop at New York in which the techniques and results bounding the strength of Infinite Time Register Machines were suggested and conjectured.

A standard TURING *computation. Head positions are indicated by shading.*

The *ordinal numbers*

$$0<1<2<3<...n<n+1<...$$
$$<\omega<\omega+1<...<\omega+n<...<\omega+\omega=\omega\cdot 2<\omega+\omega+1<\omega\cdot 2+1<...$$
$$<\omega\cdot\omega=\omega^2<...<\omega^3<...<\omega^\omega<...<\omega^{\omega^2}<...$$
$$<\omega^{\omega^{\omega^{\omega^{\cdot^{\cdot}}}}}<...<\alpha<\alpha+1<...$$
$$<\aleph_1<\aleph_1+1<...<\aleph_2<...<\aleph_3<...<\aleph_\omega<...<...$$

were introduced by GEORG CANTOR to extend the natural numbers $\mathbb{N}$ into the transfinite: ω is the smallest ordinal which follows the natural numbers $0, 1, 2, ...$; $\alpha+1$ is the immediate successor of α. The class of all ordinals is called Ord.

There are strong analogies between the ordinal structure $(\mathrm{Ord}, <, +, \cdot, 0, 1)$ and $(\mathbb{N}, <, +, \cdot, 0, 1)$:

- $(\mathrm{Ord}, <)$ is a wellorder, i.e., it is a strict linear order and every nonempty subset has a $<$-minimal element;
- ordinal addition and multiplication satisfy recursive laws familiar from natural number arithmetic.

On the other hand the ordinal line contains *limit ordinals* λ, i.e.,

$$0<\lambda\wedge\forall\alpha<\lambda\exists\beta<\lambda\colon\alpha<\beta.$$

Examples are given by ω, $\omega+\omega$, or $\omega\cdot\omega$ and by all infinite cardinals $\aleph_0, \aleph_1, \aleph_2, ...$.

An obvious generalisation from the perspective of transfinite ordinal theory is to replace the natural numbers in computations by ordinal numbers. For the TURING machine model this means working on a TURING tape indexed by *ordinals* along a time indexed by *ordinals*. At successor ordinals the *ordinal machine* behaves much like a standard finite time and finite space machine. The behaviour at limit ordinals is governed by specific *limit rules*. Limit rules are derived from the usual limit operations in the ordinals: if $(\alpha_s)_{s<\lambda}$ is a sequence of ordinals whose length λ is a limit ordinal, define its *limit* and *limit inferior* by

$$\lim_{s\to\lambda}\alpha_s=\bigcup_{s<\lambda}\alpha_s$$

and

$$\liminf_{s\to\lambda}\alpha_s=\bigcup_{s<\lambda}\bigcap_{s<r<\lambda}\alpha_r\,.$$

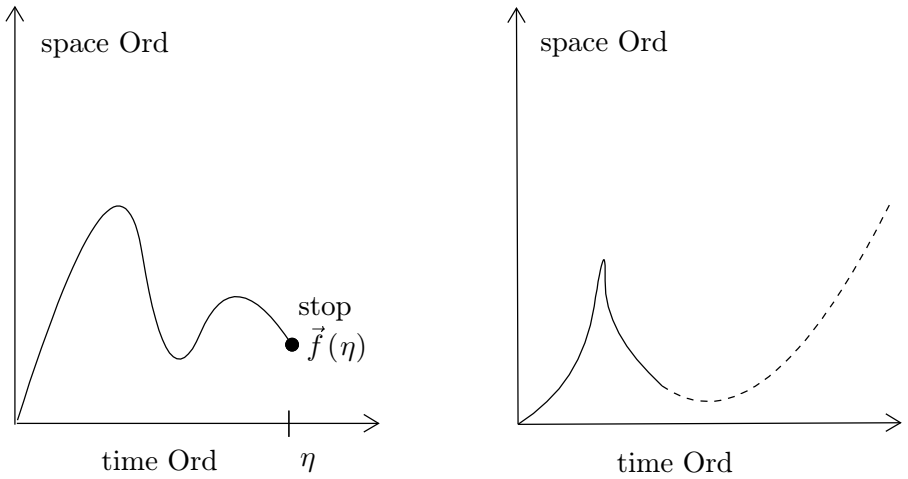
An *ordinal computation* can then be visualised like

	⋮														⋰	
↑	⋮												⋮	⋰		
	λ	0	0	0	0			0	0		0		0	⋯		
O	⋮															
R	$\beta+1$										1					
D	β	0	0	0	0			0	0		1					
I	⋮															
N	⋮				⋮		⋮			⋰						
A	$\omega+1$	0	0	0	0	0	⋯	0	0	⋯						
L	ω	0	0	0	0	0	⋯	0	1	⋯						
	⋮	⋮	⋮	⋮	⋮	⋮	⋰	⋮	⋮							
S	4	1	1	1	1	1	⋯	1	1							
P	3	1	1	1	1	1	⋯	1	1							
A	2	0	0	0	1	1	⋯	1	1							
C	1	0	0	1	1	0	⋯	0	0							
E	0	1	0	0	0	0	⋯	0	0							
		0	1	2	3	4	⋯	ω	$\omega+1$	⋯	α	⋯	κ	⋯	⋯	⋯
		O	R	D	I	N	A	L		T	I	M	E		→	

An ordinal computation.

The limit rules will be applied independently of the machine state. Thus an ordinal algorithm will be given by a *standard* program for a TURING, register or similar machine.

More abstractly, ordinal computations have the following "shapes":

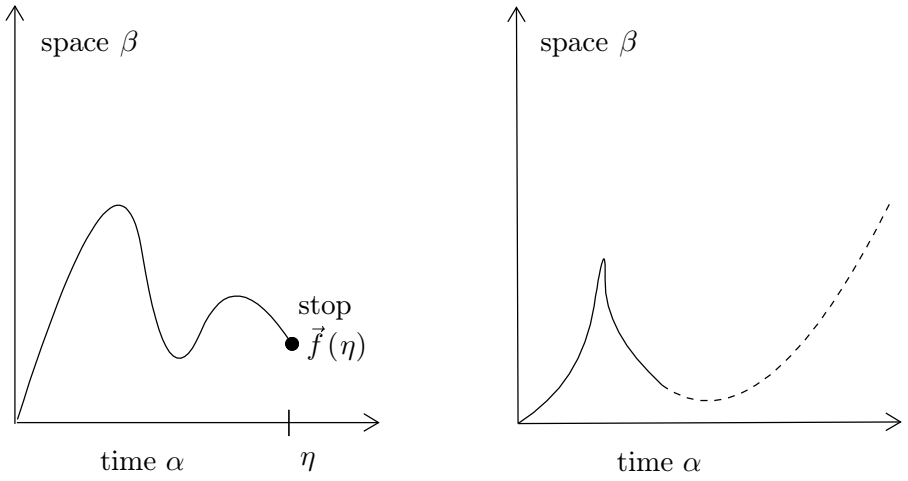


The left diagram represent a *halting* computation which computes some ordinal values $\vec{f}(\eta)$ at the halting time η. The right computation does not halt in the available time and thus *diverges*.

These diagrams suggest to consider restrictions of the available time and space by some ordinals α and β, resp. Note that in the common VON NEUMANN formalisation of ordinals, an ordinal is the set of its predecessors:

$$\alpha = \{\xi \,|\, \xi \in \alpha\} = \{\xi \,|\, \xi < \alpha\}.$$

An α-β-machine has computation diagrams of the forms:



This gives a spectrum of α-β-machines for $\omega \leqslant \alpha \leqslant \infty$ and $\omega \leqslant \beta \leqslant \infty$ where the ω-ω-machines should be equal to their standard counterparts. Here we let $\infty = \mathrm{Ord}$ be the set of all ordinals.

2 Register machines

For a concrete example we give a formal definition of register machines working on ordinals. We base our presentation of infinite time machines on the *unlimited register machines* as presented in [1]. Ordinal programs are *standard* programs.

Definition 1. *Fix limit ordinals α and β, $\omega \leqslant \alpha \leqslant \infty$, $\omega \leqslant \beta \leqslant \infty$. Let $P = I_0, I_1, ..., I_{s-1}$ be an* URM *program, i.e., a sequence of instructions as described in (i)-(v) below. Let Z: $\beta \to 2$, which will serve as an oracle. A pair*

$$I\colon \theta \to \omega, R\colon \theta \to ({}^{\omega}\beta)$$

is an α-β-(register) computation *by P if the following hold:*

a) θ is an ordinal $\leqslant \alpha$; θ is the length *of the computation;*

b) $I(0) = 0$; the machine starts in state 0;

c) If $t < \theta$ and $I(t) \notin s = \{0, 1, ..., s-1\}$ then $\theta = t + 1$; the machine stops *if the machine state is not a program state of P;*

d) If $t < \theta$ and $I(t) \in \mathrm{state}(P)$ then $t + 1 < \theta$; the next configuration is determined by the instruction $I_{I(t)}$:

i. if $I_{I(t)}$ is the instruction reset(Rn) *then let $I(t+1) = I(t) + 1$ and define $R(t+1)\colon \omega \to \mathrm{Ord}$ by*

$$R_k(t+1) = \begin{cases} 0, \text{ if } k = n \\ R_k(t), \text{ if } k \neq n \end{cases}$$

ii. *if* $I_{I(t)}$ *is the instruction* increase(Rn) *then let* $I(t+1) = I(t)+1$ *and define* $R(t+1)\colon \omega \to \mathrm{Ord}$ *by*

$$R_k(t+1) = \begin{cases} R_k(t)+1, \text{ if } k = n \\ R_k(t), \text{ if } k \neq n \end{cases}$$

iii. *if* $I_{I(t)}$ *is the oracle instruction* oracle(Rn) *then define* $R(t+1)\colon \omega \to \mathrm{Ord}$ *by*

$$R_k(t+1) = \begin{cases} Z(R_n(t)), \text{ if } k = n \\ R_k(t), \text{ if } k \neq n \end{cases}$$

iv. *if* $I_{I(t)}$ *is the transfer instruction* Rn=Rm *then let* $I(t+1) = I(t)+1$ *and define* $R(t+1)\colon \omega \to \mathrm{Ord}$ *by*

$$R_k(t+1) = \begin{cases} R_m(t), \text{ if } k = n \\ R_k(t), \text{ if } k \neq n \end{cases}$$

v. *if* $I_{I(t)}$ *is the jump instruction* if Rm=Rn then goto q *then let* $R(t+1) = R(t)$ *and*

$$I(t+1) = \begin{cases} q, \text{ if } R_m(t) = R_n(t) \\ I(t)+1, \text{ if } R_m(t) \neq R_n(t) \end{cases}$$

e) *If* $t < \theta$ *is a limit ordinal, then*

$$\begin{aligned} \forall k \in \omega (R_k(t) &= \begin{cases} \liminf_{r \to t} R_k(r) \text{ if } \liminf_{r \to t} R_k(r) < \beta \\ 0 \text{ if } \liminf_{r \to t} R_k(r) = \beta \end{cases} \\ I(t) &= \liminf_{r \to t} I(r). \end{aligned}$$

By the second clause in the definition of $R_k(t)$ *the register is* reset *in case* $\liminf_{r \to t} R_k(r) = \beta$.

The register computation is obviously determined recursively by the initial register contents $R(0)$*, the oracle* Z *and the program* P*. We call it the* α-β-register computation by P with input $R(0)$ *and oracle* Z*. If the computation stops then* $\theta = \eta + 1$ *is a* successor *ordinal and* $R(\eta)$ *is the final register content. In this case we say that* P α-β-register computes $R(\eta)(0)$ *from* $R(0)$ *and the oracle* Z*, and we write*

$$P\colon R(0), Z \mapsto_{\alpha,\beta} R(\eta)(0).$$

The interpretation of programs yields associated notions of computability.

Definition 2. *An* n*-ary function* $F\colon \omega^n \to \omega$ *is* α-β-register computable in the oracle Z *if there is a register program* P *such that for every* n*-tuple* $(a_0, ..., a_{n-1}) \in {}^{\omega}\omega$ *holds:*

$$P\colon (a_0, ..., a_{n-1}, 0, 0, ...), Z \mapsto_{\alpha,\beta} F(a_0, ..., a_{n-1}).$$

We then denote F *by* P^Z*.* F *is* α-β-register computable *if* F *is* α-β*-computable in the empty oracle* $\emptyset$*.*

Obviously any standard recursive function is α-β-computable.

Definition 3. *A subset* $A \subseteq \mathcal{P}(\beta)$ *is* α-β-register computable *if there is a register program* P*, and an oracle* $Y\colon \beta \to 2$ *such that for all* $Z \subseteq \omega$*:*

$$P^{Y \times Z}(0) = \chi_A(Z)$$

where $Y \times Z$ is some appropriate pairing of Y and Z and χ_A is the characteristic function of A.

In sections 4 and 5 we shall determine the computational strength of ∞-ω-register machines. Earlier the author studied a weaker *non-resetting* type of infinitary register machine, which was there called "Infinite Time Register Machine" [4]. Those machines exactly corresponded to hyperarithmetic definitions. Since the ∞-β-machines introduced above are in closer analogy with the established Infinite Time TURING Machines (ITTM) of [2] we use the term *Infinite Time Register Machine* for the new concept. The machines of [4] could be called *Non-Resetting Infinite Time Register Machines*.

Definition 4. *Call an ∞-ω-register machine an* Infinite Time Register Machine (ITRM). *Correspondingly we use the terms* "ITRM-computation" *and* "ITRM-computable" *for "∞-ω-register computation" and "∞-ω-register computable".*

3 Strengths of ordinal machines

An obvious task is determine the class of α-β-computable functions and sets for varying α and β. For register machines as defined above one obtains the following table. Note that one should naturally have at least as much time available as there is space ($\alpha \geqslant \beta$), to be able to reach all possible register contents. There are still open positions ("?") and ongoing research. References behind the strength statements are to the bibliography.

Ordinal register computability

Register machines	space ω	space admissible α	space Ord
time ω	standard register machine computable = Δ^0_1	-	-
time admissible α	?	α register machine (α recursion theory) computable = $\mathbf{\Delta}_1(L_\alpha)$ [6]	-
time Ord	ITRM Infinite time register machine computable = L_{ω_ω} [see sections 4 and 5]	?	Ordinal register machine computable = $L \cap \mathcal{P}(\text{Ord})$ [7]

For TURING machines we get a nearly identical table with a marked difference between ITRMs and ITTMs.

Ordinal TURING computability

TURING	space ω	space admissible α	space Ord
time ω	standard TURING machine computable $= \Delta^0_1$	-	-
time admissible α	?	α TURING machine (α-recursion theory) computable $= \mathbf{\Delta}_1(L_\alpha)$ [6]	-
time Ord	ITTM $\mathbf{\Delta}^1_1 \subsetneq$ computable in real parameter $\subsetneq \mathbf{\Delta}^1_2$ [2]	?	Ordinal TURING machine computable $= L \cap \mathcal{P}(\mathrm{Ord})$ [3]

4 Halting times of ITRMs

Determinations of α-β-computability strengths typically involve the combination of techniques from recursion theory, admissibility theory, descriptive set theory, and set theoretic constructibility theory. As an example and a new result we prove the estimate for ITRMs in detail. One can view the proofs as an analysis of the computable power of "resetting registers" at infinite liminf's: a resetting register basically corresponds to one hyperjump.

Let ω_0^{CK}, ω_1^{CK}, ..., $\omega_\omega^{\mathrm{CK}}$, ... be the monotone enumeration of the admissible ordinals and their limits. We shall prove:

Theorem 5. *A real number $a \in {}^\omega 2$ is computable by an ITRM iff $a \in L_{\omega_\omega^{\mathrm{CK}}}$.*

The implication from left to right, i.e., the upper bound for the set of ITRM-computable reals, follows from bounding the halting times, i.e., the lengths of halting computations, of ITRMs below $\omega_\omega^{\mathrm{CK}}$.

Theorem 6. *Consider an ITRM with register program P. Then there is some $n < \omega$ such that for arbitrary inputs $i < \omega$: if the ITRM-computation according to P with input i and empty oracle halts then it halts before ω_n^{CK}.*

For the proof fix an infinite time register computation $\mathcal{C}$

$$I\colon \theta \to \omega\,,\ R\colon \theta \to {}^\omega\omega$$

by P with some input $(R_0(0), R_1(0), ...)$ and oracle $\emptyset$. Assume that P only mentions registers among $R_0, ..., R_{l-1}$.

Lemma 7. *Let $\tau < \theta$ be of the form $\tau = \tau_0 + \delta$ where α is admissible $> \omega$. Assume that*

$$\forall k < l\ R_k(\tau) = \liminf_{t<\tau} R_k(t).$$

Then $\theta = \infty$.

Proof. Since computations can be composed by concatenation, we may assume that $\tau_0 = 0$ and $\tau = \delta$. Let $I(\delta) = n$ and

$$R(\delta) = (n_0, ..., n_{l-1}, 0, 0, ...)$$

where $R_0, ..., R_{l-1}$ includes all the registers mentioned in the program P. Since the constellation at δ is determined by liminf's there is some $\gamma < \delta$ such that

- $\forall t \in (\gamma, \delta)\, I(t) \geqslant n$ and $\{t \in (\gamma, \delta) | I(t) = n\}$ is closed unbounded in δ;
- for all $k < l$: $\forall t \in (\gamma, \delta)\, R_k(t) \geqslant n_k$ and $\{t \in (\gamma, \delta) | R_k(t) = n_k\}$ is closed unbounded in δ.

These closed unbounded sets are Δ_1-definable over the set L_δ. By the admissibility of L_δ their intersection is closed unbounded in δ of ordertype δ. In particular one can choose $\bar{\delta} \in (\gamma, \delta)$ such that $(I(\bar{\delta}), R_0(\bar{\delta}), ..., R_{l-1}(\bar{\delta})) = (I(\delta), R_0(\delta), ..., R_{l-1}(\delta))$ and

$$\forall t \in [\bar{\delta}, \delta]\, (I(\bar{\delta}) \leqslant I(t) \wedge R_0(\bar{\delta}) \leqslant R_0(t) \wedge ... \wedge R_{l-1}(\bar{\delta}) \leqslant R_{l-1}(t)).$$

Then one can easily show by induction, using the liminf rules: If $\sigma \geqslant \bar{\delta}$ is of the form $\sigma = \bar{\delta} + \delta \cdot \xi + \eta$ with $\eta < \delta$ then

$$(I(\sigma), R_0(\sigma), ..., R_{l-1}(\sigma)) = (I(\bar{\delta} + \eta), R_0(\bar{\delta} + \eta), ..., R_{l-1}(\bar{\delta} + \eta)).$$

In particular the computation does not stop. □

Lemma 8. *Let $n < \omega$. Let $\tau < \theta$ be of the form $\tau = \bar{\tau} + \omega_{n+1}^{\mathrm{CK}}$ and*

$$\mathrm{card}\{k < l | R_k(\tau) = 0\} \leqslant n.$$

Then $\theta = \infty$.

Proof. Set $N = \{k < l | R_k(\tau) = 0\}$. We prove the Lemma by induction on n. If $n = 0$ then $\forall k < l\, R_k(\tau) = \mathrm{liminf}_{t<\tau} R_k(t)$. By Lemma 7, $\theta = \infty$.

Now consider $n = m + 1$ where the claim holds for m. Assume that $\theta < \infty$. By Lemma 5, there must be some $k_0 < l$ such that $R_{k_0}(\tau) \neq \mathrm{liminf}_{t<\tau} R_{k_0}(t)$. By the limit rule, $R_{k_0}(\tau) = 0$ and $\mathrm{liminf}_{t<\tau} R_{k_0}(t) = \omega$. Take $\delta < \tau$ such that

$$\forall \eta \in (\delta, \tau) \forall i \in (l \setminus N) \cup \{k_0\} \colon R_i(\eta) \neq 0.$$

Take $\tau_0 \in (\delta, \tau)$ of the form $\tau_0 = \bar{\tau}_0 + \omega_n^{\mathrm{CK}} = \bar{\tau}_0 + \omega_{m+1}^{\mathrm{CK}}$. Then

$$\mathrm{card}\{k < l | R_k(\tau_0) = 0\} \leqslant n - 1 = m.$$

But then by the induction hypothesis, $\theta = \infty$, contradiction. □

We get the following corollaries:

Theorem 9. *Assume that $\theta \geqslant \omega_{l+1}^{\mathrm{CK}} + 1$, where l is the number of registers mentioned in the program P. Then $\theta = \infty$.*

Theorem 10. *If a real number $a \in {}^{\omega}2$ is computable by an ITRM then $a \in L_{\omega_\omega^{\mathrm{CK}}}$.*

Proof. The computations of $a(n)$ for $n < \omega$ have lengths $< \omega_{l+1}^{\mathrm{CK}} + 1$ for some fixed $l < \omega$. They can thus carried out absolutely *inside* the structure $(L_{\omega_\omega^{\mathrm{CK}}}, \in)$. Hence $a \in L_{\omega_\omega^{\mathrm{CK}}}$. □

5 Hyperjumps

The lower bound for the strengths of ITRM computability uses the characterization of the reals in $L_{\omega_\omega^{\mathrm{CK}}}$ by the finite *hyperjumps* $0, 0^+, 0^{++}, ..., 0^{(k+1)} = (0^{(k)})^+, ...$ of constant function 0.

Assume a fixed recursive enumeration $P_0, P_1, \ldots$ of all register programs and let $P_n^Z: \omega \rightharpoonup \omega$ be the partial function given by P with oracle Z. The hyperjump $Z^+ \in {}^\omega 2$ of $Z \in {}^\omega 2$ is defined by:

$$Z^+(n) = 1 \text{ iff } \{(i,j) \in \omega \times \omega | P_n^Z(2^i \cdot 3^j) = 1\} \text{ is a wellfounded relation.}$$

The hyperjump can be computed by the following result from [5].

Theorem 11. *The set* $\mathrm{WO} = \{Z \in {}^\omega 2 | Z \text{ codes a wellorder}\}$ *is computable by an ITRM.*

Proof. The following program P on an ITRM outputs `yes`/`no` depending on whether the oracle Z codes a wellfounded relation. The program is a backtracking algorithm which searches for a "leftmost" infinite descending chain in Z. A stack is used to organise the backtracking. We code a stack $(r_0, \ldots, r_{m-1})$ of natural numbers by $r = 2^{r_0} \cdot 3^{r_1} \cdots p_{m-1}^{r_{m-1}+1}$. We present the program in pseudo-code and assume that it is translated into a register program according to Definition 1 so that the order of commands is kept. Also the stack commands like `push` are understood as *macros* which are inserted into the code with appropriate renaming of variables and statement numbers.

```
        push 1; %% marker to make stack non-empty
        push 0; %% try 0 as first element of descending sequence
        FLAG=1; %% flag that fresh element is put on stack
Loop:   Case1: if FLAG=0 and stack=0 %% inf descending seq found
            then begin; output 'no'; stop; end;
        Case2: if FLAG=0 and stack=1 %% inf descending seq not found
            then begin; output 'yes'; stop; end;
        Case3: if FLAG=0 and length-stack > 1
        %% top element cannot be continued infinitely descendingly
            then begin; %% try next
            pop N;
            push N+1;
            FLAG:=1; %% flag that fresh element is put on stack
            goto Loop;
            end;
        Case4: if FLAG=1 and stack-is-decreasing
            then begin;
            push 0; %% try to continue sequence with 0
            FLAG:=0; FLAG:=1; %% flash the flag
            goto Loop;
            end;
        Case5: if FLAG=1 and not stack-is-decreasing
            then begin;
            pop N;
            push N+1; %% try next
            FLAG:=0; FLAG:=1; %% flash the flag
            goto Loop;
            end;
```

The correctness of this program is proved in [5]. □

From [8], Corollary VII.1.10 and the Notes for §VII.5 we obtain:

Proposition 12. *A real* $a \in {}^\omega 2$ *is an element of* $L_{\omega_\omega^{\mathrm{CK}}}$ *iff* a *is standard* TURING *computable from some* $0^{(n)}$ *with* $n < \omega$.

By Theorem 11 the collection of ITRM-computable reals is closed with respect to the hyperjump. It is also closed with respect to standard Turing computation. Proposition 12 then implies the converse direction of Theorem 5.

Theorem 13. *Every real number $a \in {}^{\omega}2 \cap L_{\omega^{\mathrm{CK}}_{\omega}}$ is computable by an ITRM.*

6 Conclusions and further considerations

Ordinal computability is able to characterise important classes of sets from higher recursion theory, descriptive set theory, and constructibility theory. It remains to be seen whether ordinal computability will have some further applications besides its unifying role. Several cases of the strengths of α-β-computability still need to be determined and are the subject of current research.

Concerning the specific result on ITRMs we plan to refine the above analysis to a level-by-level correspondence between numbers of registers, numbers of the admissible ordinals used to bound halting computations, and numbers of iterations of the hyperjump. One register should correspond to one admissible, and to one application of the hyperjump. Further registers will be required for auxiliary classical computations and bookkeeping.

Bibliography

[1] Nigel J. Cutland. *Computability: An introduction to Recursive Function Theory*. Perspectives in Mathematical Logic. Cambridge University Press, 1980.

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

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

[4] Peter Koepke. Infinite Time Register Machines. In A. Beckmann et al, editor, *Logical Approaches to Computational Barriers*, volume 3988 of *Lecture Notes in Computer Science*, pages 257–266. Springer Verlag, 2006.

[5] Peter Koepke and Russell Miller. An Enhanced Theory of Infinite Time Register Machines. In A. Beckmann et al, editor, *Logic and theory of Algorithms*, volume 5028 of *Lecture Notes in Computer Science*, pages 306–315. Springer Verlag, 2008.

[6] Peter Koepke and Benjamin Seyfferth. Ordinal machines and admissible recursion theory. *Annals of Mathematical Logic*, to appear, 2009.

[7] Peter Koepke and Ryan Siders. Register Computations on Ordinals. *Archive for Mathematical Logic*, 47:529–548, 2008.

[8] Stephen G. Simpson. *Subsystems of Second Order Arithmetic*. Perspectives in Mathematical Logic. Springer-Verlag, 1999.