Ordinal Computability

by Peter Koepke

University of Bonn

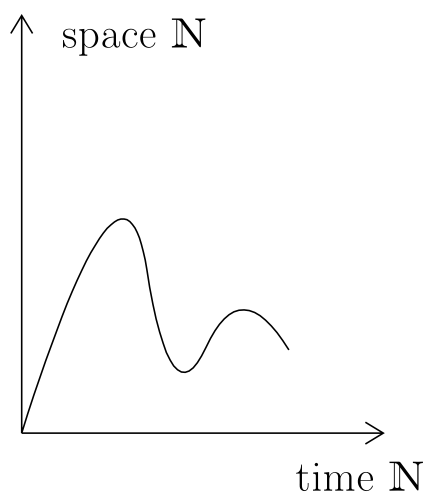
EMU - Effective Mathematics of the Uncountable
CUNY Graduate Center, August 8, 2008

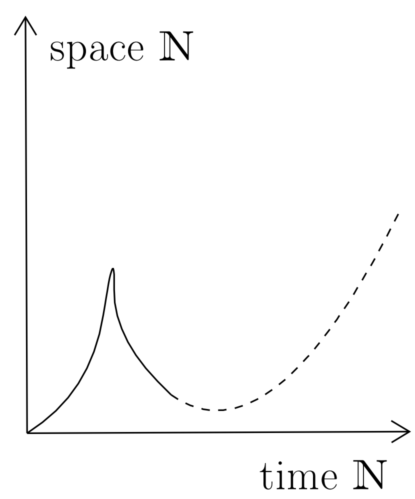
A standard Turing computation

	⋮	⋮	⋮	⋮	⋮	⋮		⋮	⋮		
	$n+1$	0	0	0	0	0		0	1	…	…
$\Uparrow$	n	0	0	0	0	0	…	1	1		
	⋮	0	0	0	0	0	…	0	0		
S	4	0	0	0	0	0	…	0	0		
P	3	0	0	0	0	0	…	0	0		
A	2	0	0	1	1	1	…	1	1		
C	1	0	1	1	0	0	…	0	0		
E	0	1	1	1	1	0	…	1	1		
		0	1	2	3	4	…	n	$n+1$	…	…

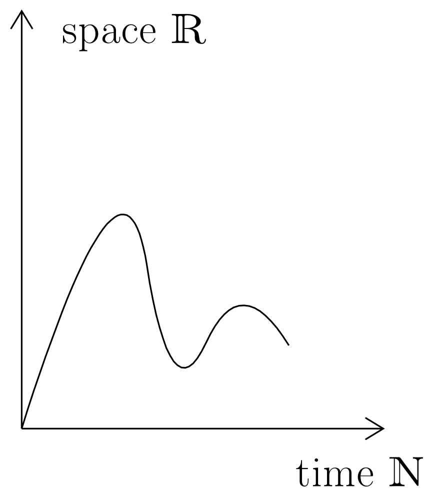
T I M E $\Rightarrow$

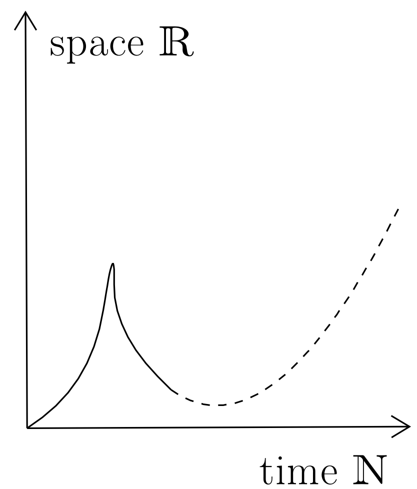
The shape of standard Turing computations



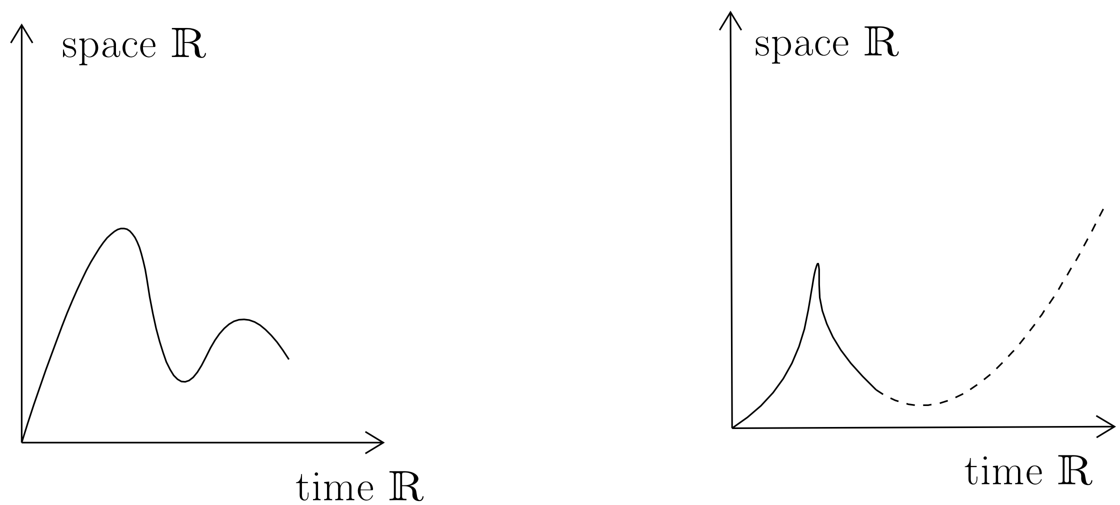


The shape of BSS computations

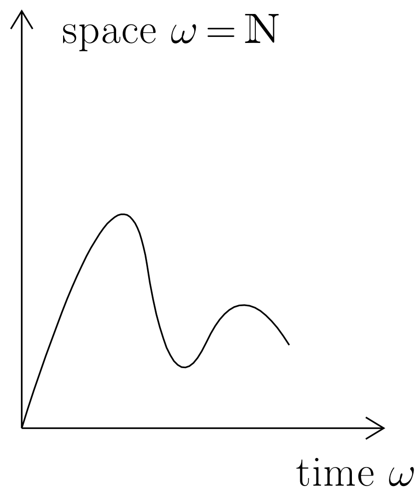


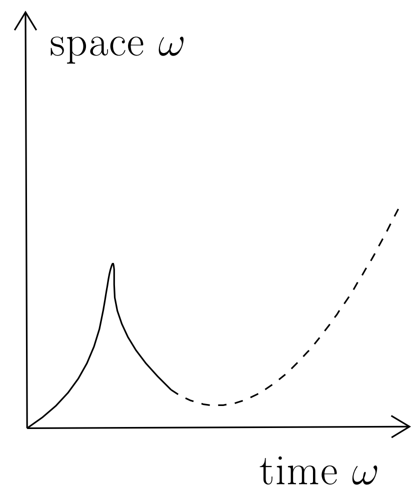


Real functions, differential equations, dynamical systems



Standard Turing computations are based on the *ordinal* $\boldsymbol{\omega} = \mathbb{N}$





Ordinals

Natural numbers:

$0 = \emptyset,\ 1 = \{0\},\ 2 = \{0,1\},\ \ldots,\ n = \{0,1,\ldots,n-1\},\ \ldots$

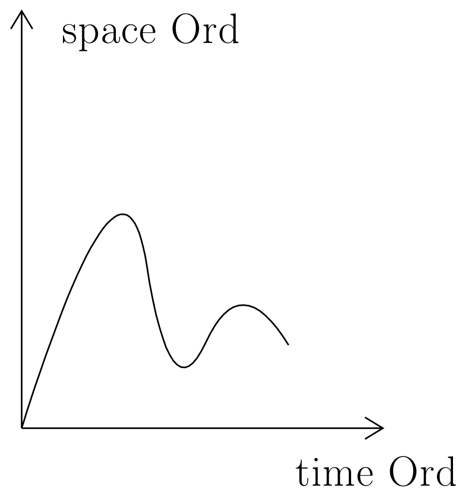
$\omega = \mathbb{N} = \{0,1,2,\ldots,n,\ldots\}$

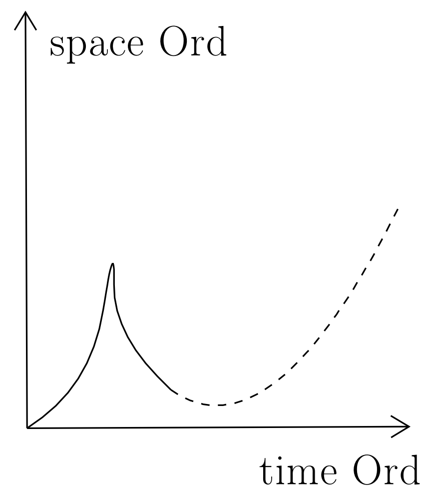
Ordinal numbers:

$0,1,2,\ldots,n,\ldots,\omega,\omega+1 = \omega \cup \{\omega\},\ldots,\alpha,\alpha+1 = \alpha \cup \{\alpha\},\ldots,\aleph_1,\ldots,\aleph_\omega,\ldots$

$\infty = \mathrm{Ord} = \{0,1,2,\ldots,\omega,\ldots,\alpha,\ldots\}$

Ordinal computations





Limit ordinals and ordinal limits

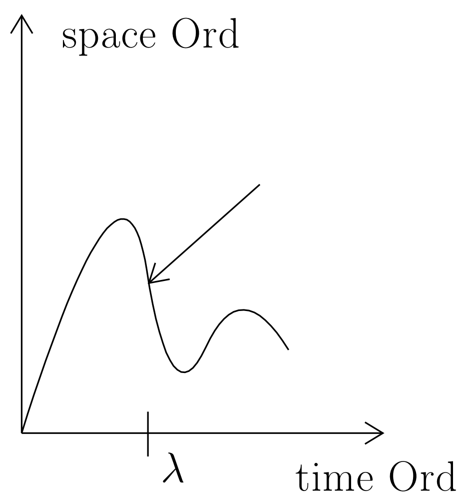
An ordinal λ is a *limit ordinal*, if it is not of the form $\lambda = 0$ or $\lambda = \mu + 1$.

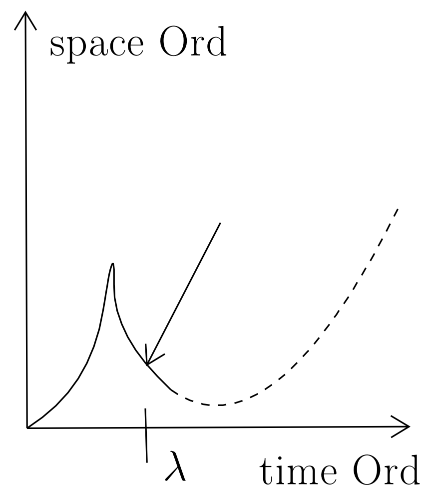
Let $\{\alpha_\xi | \xi < \lambda\} \subseteq \mathrm{Ord}$.

$\sup_{\xi<\lambda} \alpha_\xi = \bigcup_{\xi<\lambda} \alpha_\xi \in \mathrm{Ord}$, $\min_{\xi<\lambda} \alpha_\xi = \bigcap_{\xi<\lambda} \alpha_\xi \in \mathrm{Ord}$.

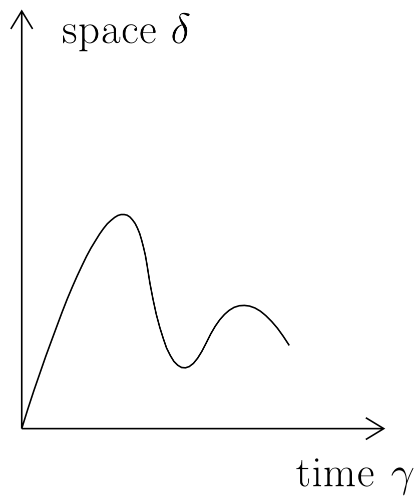
$\liminf_{\xi<\lambda} \alpha_\xi = \sup_{\zeta<\lambda} (\min_{\zeta\leqslant\xi<\lambda} \alpha_\xi)$.

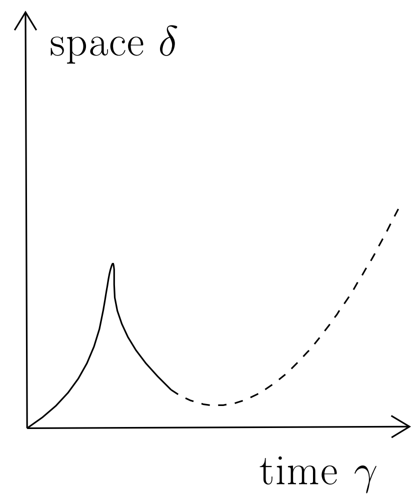
Ordinal computations: $\liminf$ at limit ordinals



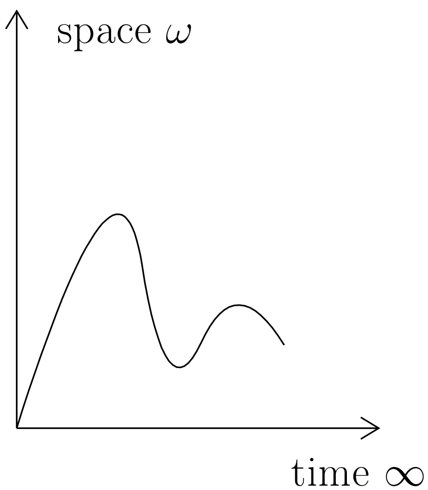


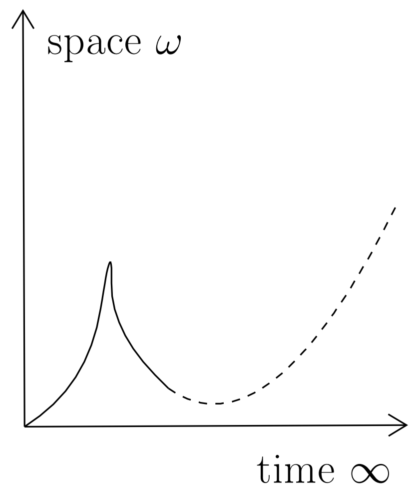
γ-δ-computations



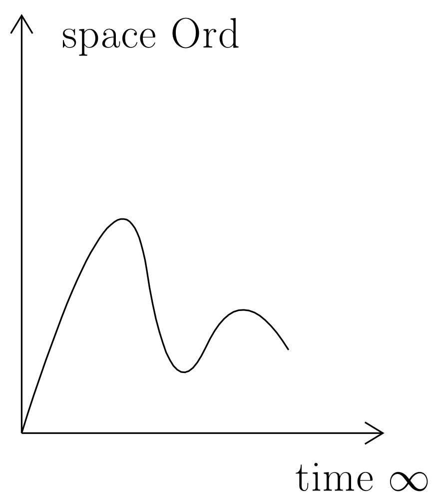


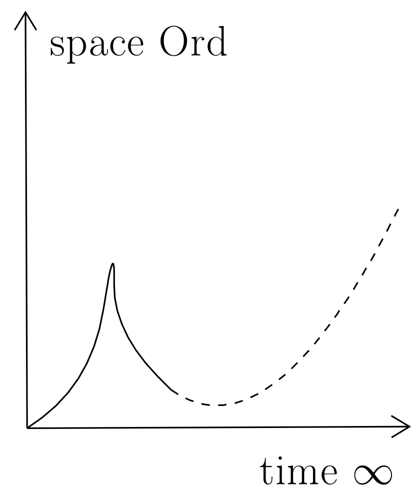
ITTM computations are ∞-ω-computations





Ordinal register machines (ORM) (with Ryan Siders)





A *register program* is a finite list $P = I_0, I_1, ..., I_{s-1}$ of *instructions*:

a) the *zero instruction* $Z(n)$ set register R_n to 0;

b) the *successor instruction* $S(n)$ increases register R_n by 1;

c) the *oracle instruction* $O(n)$ sets register R_n to 1 if its content is an element of the oracle, and to 0 otherwise;

d) the *transfer instruction* $T(m, n)$ sets R_n to the contents of R_m;

e) the *jump instruction* $J(m, n, q)$: if $R_m = R_n$, the register machine proceeds to the qth instruction of P; otherwise it proceeds to the next instruction in P.

Let $P = P_0, P_1, ..., P_{k-1}$ be a register program. A pair

$$S: \theta \to \omega, R: \theta \to ({}^{\omega}\mathrm{Ord})$$

is the ORM *computation* by P *with oracle* $Z \subseteq \mathrm{Ord}$ if:

a) θ is a successor ordinal or $\theta = \mathrm{Ord}$; θ is the *length* of the computation;

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

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

d) If $t < \theta$ and $S(t) \in \{0, 1, ..., s-1\}$ then $t + 1 < \theta$; the next configuration is determined by the instruction $P_{S(t)}$: ....

e) If $t < \theta$ is a limit ordinal, the machine constellation at t is determined by taking inferior limits:

$$\begin{aligned} \forall k \in \omega \; R_k(t) &= \liminf_{r \to t} R_k(r); \\ S(t) &= \liminf_{r \to t} S(r). \end{aligned}$$

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

$x \subseteq \mathrm{Ord}$ is ORM *computable* (from parameters) if there are a program P and ordinals $\delta_1, \ldots, \delta_{n-1}$ such that

$$\forall \alpha \; P\colon (\alpha, \delta_1, \ldots, \delta_{n-1}, 0, 0, \ldots) \mapsto \chi_x(\alpha),$$

where χ_x is the characteristic function of x.

Theorem. $x \subseteq \mathrm{Ord}$ is ORM *computable* iff $x \in L$, where L is GÖDEL's inner model of constructible sets.

Proof. $\rightarrow$ is obvious, since ORM computations can be carried out in L with the same results.

$\leftarrow$ relies on the following

Recursion Theorem. Let $H\colon \mathrm{Ord}^3 \to \mathrm{Ord}$ be ORM computable. Define

$$F(\alpha) = \begin{cases} 1 \text{ iff } \exists \nu < \alpha\ H(\alpha, \nu, F(\nu)) = 1 \\ 0 \text{ else} \end{cases}$$

Then $F\colon \mathrm{Ord} \to \mathrm{Ord}$ is ORM computable.

Proof. To determine $F(\alpha_0)$, organize the search for $\alpha_1 < \alpha_0$ with $H(\alpha_0, \alpha_1, F(\alpha_1)) = 1$ and the search for $F(\alpha_1)$ by a *stack*

$$F(\alpha_0)?, F(\alpha_1)?, \ldots, F(\alpha_{n-1})?$$

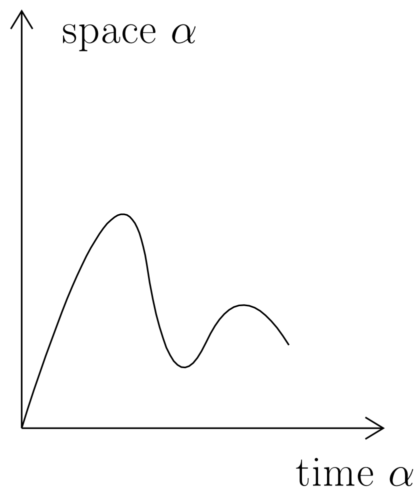
Code the stack $\alpha_0 > \alpha_1 > \ldots > \alpha_{n-1}$ by one ordinal

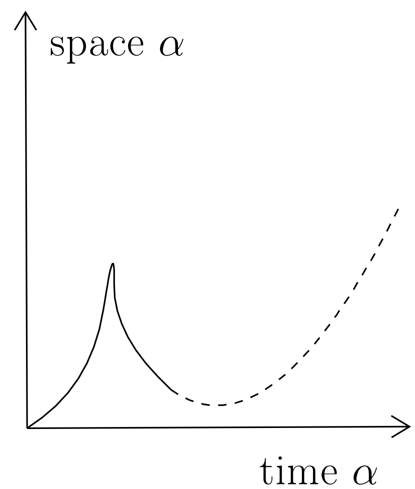
$$\alpha = \langle \alpha_0, \alpha_1, \ldots, \alpha_{n-2}, \alpha_{n-1} \rangle = 3^{\alpha_0} + 3^{\alpha_1} + \ldots + 3^{\alpha_{n-2}} + 3^{\alpha_{n-1}}.$$

```
value:=2
MainLoop:
  nu:=last(stack)
  alpha:=llast(stack)
  if nu = alpha then
1:  do
   remove_last_element_of(stack)
    value:=0
    goto SubLoop
    end_do
  else
2:  do
    stack:=stack + 1
    goto MainLoop
    end_do

SubLoop:
  nu:=last(stack)
  alpha:=llast(stack)
  if alpha = UNDEFINED then STOP
  else
    do
    if H(alpha,nu,value)=1 then
3:    do
   remove_last_element_of(stack)
      value:=1
      goto SubLoop
      end_do
    else
4:    do
      stack:=stack + 2*(3**y)
      value:=2

      goto MainLoop
      end_do
    end_do
```

- computable approach to L
- proving the continuum hypothesis = counting the number of ORM computable subsets of ω
- fine structure of L: define SILVER machines from an ORM program which "computes L"
- are (some) fine structural constructions computations?
- approximate ∞-∞-machines by α-α-machines, $\alpha \rightarrow \infty$

$\boldsymbol{\alpha}$-$\boldsymbol{\alpha}$-computations for admissible $\boldsymbol{\alpha}$ (with Benjamin Seyfferth)





Theorem. Let α be an admissible ordinal and $X \subseteq \alpha$. Then

a) X is computable by an α-α-register machine in parameters $< \alpha$ iff $X \in \mathbf{\Delta}_1^1(L_\alpha)$

b) X is computably enumerable by an α-α-register machine in parameters $< \alpha$ iff $X \in \mathbf{\Sigma}_1^1(L_\alpha)$

One can characterize when a limit ordinal β is admissible using β-β-machines.
One can do parts of α recursion theory using α-α-machines, e.g., the SACKS-SIMPSON theorem.

Ordinal register computability

Register machines	space ω	space admissible α	space Ord
time ω	standard register machine computable $= \Delta_1^0$	-	-
time admissible α	?	α register machine (α recursion theory) computable $= \mathbf{\Delta}_1(L_\alpha)$	-
time Ord	?	?	Ordinal register machine computable $= L \cap \mathcal{P}(\mathrm{Ord})$

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 $= \mathbf{\Delta}_1(L_\alpha)$	-
time Ord	ITTM $\mathbf{\Delta}^1_1 \subsetneq$ computable in real parameter $\subsetneq \mathbf{\Delta}^1_2$	?	Ordinal Turing machine computable $= L \cap \mathcal{P}(\mathrm{Ord})$

Ordinal register computability

Register machines	space ω	space admissible α	space Ord
time ω	standard register machine computable $= \Delta^0_1$	-	-
time admissible α	?	α register machine (α recursion theory) computable $= \mathbf{\Delta}_1(L_\alpha)$	-
time Ord	ITRM Infinite time register machine computabel in real parameters = ?	?	Ordinal register machine computable $= L \cap \mathcal{P}(\mathrm{Ord})$

Infinite Time Register Machines (ITRM) (with Russell Miller)

Let $P = P_0, P_1, \ldots, P_{k-1}$ be a register program. A pair

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

is the *infinite time register computation* by P *with oracle* $Z \subseteq \omega$ if:

a) ...

b) If $t < \theta$ is a limit ordinal, the machine constellation at t is determined by taking inferior limits **or in case of overflow resetting to** 0:

$$\begin{aligned} \forall k \in \omega \ R_k(t) &= \begin{cases} 0, \text{ if } \liminf_{r \to t} R_k(r) = \omega, \\ \liminf_{r \to t} R_k(r), \text{ else;} \end{cases} \\ S(t) &= \liminf_{r \to t} S(r). \end{aligned}$$

A subset $A \subseteq \mathcal{P}(\omega) = \mathbb{R}$ is ITRM-*computable* if there is a register program P and an oracle $Y \subseteq \omega$ such that for all $Z \subseteq \omega$:

$$Z \in A \text{ iff } P\colon (0, 0, \ldots), Y \times Z \mapsto 1, \text{ and } Z \notin A \text{ iff } P\colon (0, 0, \ldots), Y \times Z \mapsto 0$$

where $Y \times Z$ is the cartesian product of Y and Z with respect to the pairing function

$$(y, z) \mapsto \frac{(y + z)(y + z + 1)}{2} + z.$$

Stacks

Code a stack $(r_0, ..., r_{m-1})$ of natural numbers by

$$r = 2^m \cdot 3^{r_0} \cdot 5^{r_1} \cdots p_m^{r_{m-1}}$$

Proposition 1. *Let $\alpha < \tau$ where τ is a limit ordinal. Assume that in some ITRM-computation using a stack, the stack contains $r = (r_0, ..., r_{m-1})$ for cofinally many times below τ and that all contents in the time interval (α, τ) are endextensions of $r = (r_0, ..., r_{m-1})$. Then at time τ the stack contents are*

$$r = (r_0, ..., r_{m-1}).$$

```
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
          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;
```

Lemma 2. *Let* $I\colon \theta \to \omega$, $R\colon \theta \to ({}^{\omega}\omega)$ *be the computation by* P *with oracle* Z *and trivial input* $(0,0,\ldots)$. *Then*

a) *If* Z *is wellfounded then the computation stops with output* `'yes'`.

b) *If* Z *is illfounded then the computation stops with output* `'no'`.

Theorem 3. *The set* $\mathrm{WO} = \{Z \subseteq \omega \,|\, Z \text{ codes a wellorder}\}$ *is computable by an ITRM.*

Theorem 4. *Every* Π^1_1 *set* $A \subseteq \mathcal{P}(\omega)$ *is ITRM-computable.*

ITTMs can simulate ITRMs:

Simulate the number i in register R_m as an initial segment of i 1's on the m-th tape of an ITTM.

If λ is a limit time and $\liminf_{\tau\to\lambda} R_m(\tau) = i^* \leqslant \omega$ then the m-th tape will hold an initial segment of i^* 1's.

OK, if i^* is finite.

If $i^* = \omega$, this may be detected by a subroutine which then *resets* the m-th tape to 0.

Since ITTM-decidable $\subseteq \mathbf{\Delta}^1_2$:

Ordinal register computability

Register machines	space ω	space admissible α	space Ord
time ω	standard register machine computable $= \Delta_1^0$	-	-
time admissible α	?	α register machine (α recursion theory) computable $= \boldsymbol{\Delta}_1(L_\alpha)$	-
time Ord	ITRM $\boldsymbol{\Delta}_1^1 \subsetneq$ computable in real parameter $\subsetneq \boldsymbol{\Delta}_2^1$	?	Ordinal register machine computable $= L \cap \mathcal{P}(\mathrm{Ord})$

ITRMs, ITTMs, and halting problems

(S, R) is a *configuration* if $S \in \omega$ is a program state and $R\colon \omega \to \omega$ where $R(n) = 0$ for almost all $n < \omega$. Define a wellfounded partial order of configurations

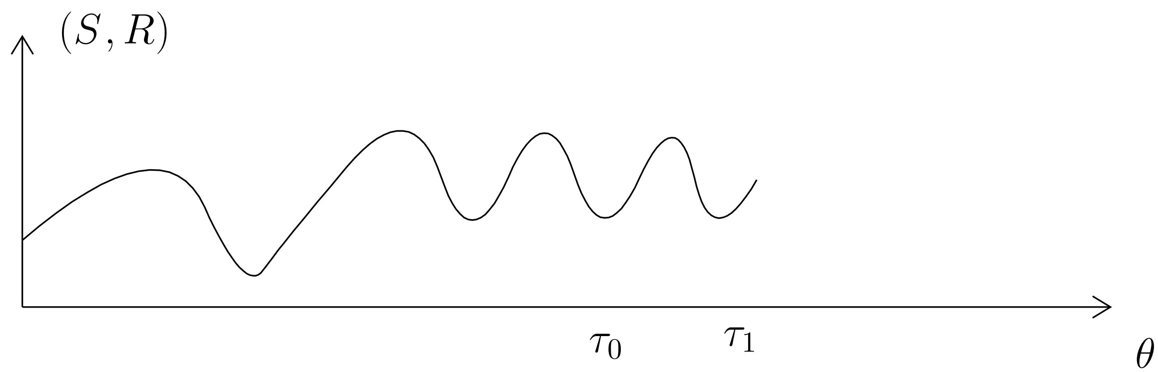
$$(S_0, R_0) \leqslant (S_1, R_1) \text{ iff } S_0 \leqslant S_1 \text{ and } \forall n < \omega\, R_0(n) \leqslant R_1(n).$$

Lemma 5. *Let*

$$S: \theta \rightarrow \omega, R: \theta \rightarrow ({}^{\omega}\omega)$$

be the infinite time register computation by P with input $(0, 0, \ldots)$ *and oracle Z. Then this computation does* not *halt iff there are $\tau_0 < \tau_1 < \theta$ such that*

$$(S(\tau_0), R(\tau_0)) = (S(\tau_1), R(\tau_1)) \text{ and } \forall \tau \in [\tau_0, \tau_1] (S(\tau_0), R(\tau_0)) \leqslant (S(\tau), R(\tau)).$$



Proof. $(\rightarrow)$ Assume that the computation does not halt. Let A be the set of all configurations occuring class-many times. A is downwards directed in the partial order of configurations:

for $(S_0, R_0), (S_1, R_1) \in A$ choose a sufficiently high ω-sequence $\tau_0 < \tau_1 < \cdots$ of stages such that each (S_i, R_i) occurs at all stages of the form $\tau_{2 \cdot k + i}$ with $i < 2$.

Then (S, R) occuring at stage $\sup_n \tau_n$ has $(S, R) \leqslant (I_0, R_0)$ and $(S, R) \leqslant (I_1, R_1)$.

Let (S_0, R_0) be the unique $\leqslant$-minimal element of A. Choose sufficiently high stages τ_0, τ_1 such that $\tau_0 < \tau_1$ with $(S(\tau_0), R(\tau_0)) = (S(\tau_1), R(\tau_1)) = (S_0, R_0)$.

$(\leftarrow)$ For the converse assume that there are $\tau_0 < \tau_1 < \theta$ such that and

$$(S(\tau_0), R(\tau_0)) = (S(\tau_1), R(\tau_1)) \text{ and } \forall \tau \in [\tau_0, \tau_1]\, (S(\tau_0), R(\tau_0)) \leqslant (S(\tau), R(\tau)).$$

Then if $\sigma \geqslant \tau_0$ is of the form $\sigma = \tau_0 + (\tau_1 - \tau_0) \cdot \alpha + \beta,\ \beta < \tau_1 - \tau_0$ then

$$(S(\sigma), R(\sigma)) = (S(\tau_0 + \beta), R(\tau_0 + \beta)).$$

So the computation does not stop. □

Theorem 6. *The* halting problem *for ITRMs*

$$\{(P,Z) \mid P \textit{ is a register program, } Z \subseteq \omega, \textit{ and the computation by } P \textit{ with input } (0,0,\ldots) \textit{ and oracle } Z \textit{ halts}\}$$

is decidable by an ITTM with oracle Z.

ITRMs are weaker than ITTMs.

Proof. Implement the criterion of Lemma 5 on an ITTM.

Simulate the computation for (P,Z).

Use an auxiliary tape with cells for each possible configuration of the ITRM.

At stage τ of the simulation erase from the auxiliary tape all 1's for configurations which are not $\leqslant (S(\tau), R(\tau))$, put a 1 for the configuration $(S(\tau), R(\tau))$.

If there was already a 1 in this cell, then by Lemma 5 the computation diverges.

If the simulation stops the computation stops. □

Theorem 7. *The* restricted halting problem *for ITRMs*

$$\{(P,Z) \mid P \textit{ is a register program using at most } N \textit{ registers, } Z \subseteq \omega, \\ \textit{and the computation by } P \textit{ with input } (0,0,\ldots) \textit{ and oracle } Z \textit{ halts}\}$$

is decidable by an ***ITRM*** *with oracle Z, for every* $N < \omega$.

Proof. Emulate the bookkeeping of the previous proof using auxiliary registers.

$$C(\tau) = \{(S(\sigma), R(\sigma)) | \sigma < \tau \wedge \forall \sigma' \in [\sigma, \tau] (S(\sigma), R(\sigma)) \leqslant (S(\sigma'), R(\sigma'))\}$$

The halting criterion becomes

$$\exists \tau\, ((S(\tau), R(\tau)) \in C(\tau)).$$

$C(\tau)$ can be carried along using $N + \mathrm{const}$ extra registers. □

Theorem 8.

The strength of ITRMs using N registers grows eventually strictly with N.

There cannot be a universal ITRM.

Question. For which N is an N-register ITRM strictly weaker than an $N+1$-register ITRM?

Infinite time register computable model theory

Follow Hamkins, Miller, Seabold, Warner *Infinite Time Computable Model Theory* using:

- decide WF and WO
- decide the elementary diagram of first-order structures on $\mathbb{N}$ since it is Δ_1^1 in the code for the structure
- *lost melody theorem*