Archive for Mathematical Logic manuscript No.
(will be inserted by the editor)

Register Computations on Ordinals

Peter Koepke · Ryan Siders

the date of receipt and acceptance should be inserted later

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

Keywords Ordinal computability · hypercomputation · infinitary computation · register machine

1 Introduction.

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

Peter Koepke
University of Bonn, Mathematisches Institut, Beringstraße 1, D 53115 Bonn, Germany
E-mail: koepke@math.uni-bonn.de

Ryan Siders
University of Helsinki, Department of Mathematics, P.O.Box 4, Yliopistonkatu 5, FI 00014 Helsinki, Finland
E-mail: bissell@mappi.helsinki.fi

one defines the program states and the registers contents by appropriate limit operations.

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

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

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

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

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

The class

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

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

Theorem 1 *A set $x \subseteq \mathrm{Ord}$ is ordinal computable if and only if $x \in L$.*

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

2 Ordinal register machines

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

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

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

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

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

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

Definition 2 Let $P = P_0, P_1, \ldots, P_{k-1}$ be an ORM program. A pair

$$I : \theta \to \omega, R : \theta \to {}^{\omega}\,\text{Ord}$$

is an *ordinal (register) computation* by P if the following hold:

a) θ is a successor ordinal or $\theta = \text{Ord}$; θ is the *length* of the computation;
b) $I(0) = 0$; the machine starts in state 0;
c) If $t < \theta$ and $I(t) \notin k = \{0, 1, \ldots, k-1\}$ then $\theta = t+1$; the machine *stops* if the machine state is not a program state of P;
d) If $t < \theta$ and $I(t) \in \{0, 1, \ldots, k-1\}$ then $t+1 < \theta$; the next configuration is determined by the instruction $P_{S(t)}$:
   i. if $P_{S(t)}$ is the zero instruction `Z(n)` then let $I(t+1) = I(t) + 1$ and define $R(t+1) : \omega \to \text{Ord}$ by
   $$R_k(t+1) = \begin{cases} 0, & \text{if } k = n \\ R_k(t), & \text{if } k \neq n \end{cases}$$
   ii. if $P_{S(t)}$ is the successor instruction `S(n)` then let $I(t+1) = I(t) + 1$ and define $R(t+1) : \omega \to \text{Ord}$ by
   $$R_k(t+1) = \begin{cases} R_k(t) + 1, & \text{if } k = n \\ R_k(t), & \text{if } k \neq n \end{cases}$$
   iii. if $P_{S(t)}$ is the transfer instruction `T(m,n)` then let $I(t+1) = I(t) + 1$ and define $R(t+1) : \omega \to \text{Ord}$ by
   $$R_k(t+1) = \begin{cases} R_m(t), & \text{if } k = n \\ R_k(t), & \text{if } k \neq n \end{cases}$$

iv. if $P_{S(t)}$ is the jump instruction `J(m,n,q)` then let $R(t+1) = R(t)$ and

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

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

$$\forall k \in \omega \ R_k(t) = \liminf_{r \to t} R_k(r);$$
$$I(t) = \liminf_{r \to t} I(r).$$

The ordinal computation is obviously recursively determined by the initial register contents $R(0)$ and the program P. We call it the *ordinal computation by P with imput $R(0)$*. If the computation stops, $\theta = \beta + 1$ is a successor ordinal and $R(\beta)$ is the final register content. In this case we say that P *computes* $R(\beta)(0)$ from $R(0)$ and write $P : R(0) \mapsto R(\beta)(0)$.

The definition of the state $I(t)$ for limit t can be motivated as follows. Since a program is finite its execution will lead to some (complex) looping structure involving loops, subloops and so forth. This can be presented by pseudo code like:

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

Assume that for times $r \to t$ the loop $(17-32)$ with its subloop $(21-29)$ is traversed cofinally often. Then at time t it seems natural to put the machine at the start of the "main loop". Assuming that the lines of the program are enumerated in increasing order this corresponds to the lim inf rule

$$I(t) = \liminf_{r \to t} I(r).$$

The interpretation of programs by computations yields associated notions of computability.

Definition 3 An n-ary partial function $F : \mathrm{Ord}^m \rightharpoonup \mathrm{Ord}$ is *ordinal (register) computable* if there are a register program P and ordinals $\delta_0, \ldots, \delta_{n-1}$ such that for every m-tuple $(\alpha_0, \ldots, \alpha_{m-1}) \in \operatorname{dom} F$ holds

$$P : (\alpha_0, \ldots, \alpha_{m-1}, \delta_0, \ldots, \delta_{n-1}, 0, 0, \ldots) \mapsto F(\alpha_0, \ldots, \alpha_{m-1}).$$

A subset $x \subseteq \mathrm{Ord}$ is *ordinal (register) computable* if its characteristic function χ_x is ordinal computable.

3 Algorithms

Since ordinal register machines are a straightforward extension of standard register machines, all recursive functions can be computed by an ordinal register machine. We shall now show that basic operations on ordinal numbers are ordinal register computable. We present programs in an informal *pseudo code* where variables correspond to registers.

```
Ordinal addition, computing gamma = alpha + beta:
0  alpha':=0
1  beta':=0
2  gamma:=0
3  if alpha=alpha' then go to 7
4  alpha':=alpha'+1
5  gamma:=gamma+1
6  go to 3
7  if beta=beta' then STOP
8  beta':=beta'+1
9  gamma:=gamma+1
10 go to 7
```

Observe that at limit times this algorithm, by the lim inf rule, will nicely cycle back to the beginnings of loops `3 - 6` or `7 - 10` resp.

```
Ordinal multiplication, computing gamma = alpha * beta:
0  beta':=0
1  gamma:=0
2  if beta=beta' then STOP
3  beta':=beta'+1
4  gamma:=gamma + alpha
5  go to 2
```

We interpret the program line `gamma:=gamma + alpha` as a *macro*, i.e., the above addition program has to be substituted for that line with reasonable modifications of variables, registers and line numbers. Also adequate transfer of arguments and values between variables has to be arranged.

In general this substitution technique yields the closure under composition for the class of ordinal computable functions:

Theorem 2 *Let $f(v_0, \ldots, v_{n-1})$ and $g_0(\overrightarrow{w}), \ldots, g_{n-1}(\overrightarrow{w})$ be ordinal computable functions. Then the composition $h(\overrightarrow{w}) = f(g_0(\overrightarrow{w}), \ldots, g_{n-1}(\overrightarrow{w}))$ is ordinal computable.*

The Gödel pairing function for ordinals is important for coding information into single ordinals. It is defined recursively by

$$\begin{aligned} G(\alpha,\beta) = \{G(\alpha',\beta') \mid\, & \max(\alpha',\beta') < \max(\alpha,\beta) \text{ or} \\ & (\max(\alpha',\beta') = \max(\alpha,\beta) \text{ and } \alpha' < \alpha) \text{ or} \\ & (\max(\alpha',\beta') = \max(\alpha,\beta) \text{ and } \alpha' = \alpha \text{ and } \beta' < \beta)\}. \end{aligned}$$

We sketch an algorithm for computing $\gamma = G(\alpha, \beta)$, it proceeds by increasing a pair (α', β') along the well-order of $\mathrm{Ord} \times \mathrm{Ord}$ implicit in the definition of G until (α, β) is reached and simultaneously increasing the ordinal γ along the ordinals.

```
Goedel pairing, computing gamma = G(alpha,beta):
0  alpha':=0
1  beta':=0
2  eta:=0
3  flag:=0
4  gamma:=0
5  if alpha=alpha' and beta=beta' then STOP
6  if alpha'=eta and and beta'=eta and flag=0 then
    alpha':=0, flag:=1, gamma:=gamma+1, go to 5 fi
7  if alpha'=eta and and beta'=eta and flag=1 then
    eta:=eta+1,alpha':=eta,beta':=0,gamma:=gamma+1, go to 5 fi
8  if beta'<eta and flag=0 then
    beta':=beta'+1, gamma:=gamma+1, go to 5 fi
9  if alpha'<eta and flag=1 then
    alpha':=alpha'+1, gamma:=gamma+1, go to 5 fi
```

The inverse functions G_0 and G_1 satisfying

$$\forall \gamma \gamma = G(G_0(\gamma), G_1(\gamma))$$

are also ordinal computable: compute $G(\alpha, \beta)$ for $\alpha, \beta < \gamma$ until you find α, β with $G(\alpha, \beta) = \gamma$; then set $G_0(\gamma) = \alpha$ and $G_1(\gamma) = \beta$. This is a special case of the following inverse function theorem.

Theorem 3 *Let the function $f : \mathrm{Ord}^n \to \mathrm{Ord}$ be ordinal computable and surjective. Then there are ordinal computable functions $g_0, \ldots, g_{n-1} : \mathrm{Ord} \to \mathrm{Ord}$ such that*

$$\forall \alpha f(g_0(\alpha), \ldots, g_{n-1}(\alpha)) = \alpha.$$

4 3-adic representations and ordinal stacks

We shall compute a recursive truth function using a *stack* that can hold a (finite) sequence $\alpha_0 > \alpha_1 > \ldots > \alpha_{n-2} \geqslant \alpha_{n-1}$ of ordinals which is strictly decreasing except possibly for the last two ordinals. This sequence of ordinals will be coded into a single ordinal by 3-adic representations.

Proposition 1 *Let $\delta > 1$ be a fixed* basis *ordinal. A representation*

$$\alpha = \delta^{\alpha_0} \cdot \zeta_0 + \delta^{\alpha_1} \cdot \zeta_1 + \ldots + \delta^{\alpha_{n-1}} \cdot \zeta_{n-1}$$

*with $\alpha_0 > \alpha_1 > \ldots > \alpha_{n-1}$ and $0 < \zeta_0, \zeta_1, \ldots, \zeta_{n-1} < \delta$ is called a δ-*adic representation *of α.*

It is an easy exercise in ordinal arithmetic to show that every $\alpha \in \mathrm{Ord}$ possesses a unique δ-adic representation. So a decreasing stack $\alpha_0 > \alpha_1 > \ldots > \alpha_{n-2} \geqslant \alpha_{n-1}$ of ordinals can be coded by

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

We call the natural number n the *length* of the stack α . The elements $\alpha_{n-1}, \alpha_{n-2}, \ldots$ of this stack can be defined from α as follows:

$$\begin{aligned} \alpha_{n-1} &= \text{the largest } \xi \text{ such that there is } \zeta \text{ with } \alpha = 3^{\xi} \cdot \zeta \\ \alpha_{n-2} &= \text{the largest } \xi \text{ such that there is } \zeta \text{ with } \alpha = 3^{\xi} \cdot \zeta + 3^{\alpha_{n-1}} \\ &\ldots \end{aligned}$$

Since the ordinal arithmetic operations are ordinal computable, the ordinals $\alpha_{n-1}, \alpha_{n-2}$ are ordinal computable by some programs `last`, `llast` resp. We assume that these functions return a special value `UNDEFINED` if the stack is too short.

The computation in the subsequent recursion theorem proceeds by ranging over previous arguments and values in a systematic way. This can be organized by a stack, due to the limit behaviour of stacks.

Proposition 2 *Let $t \in \mathrm{Ord}$ be a limit time and $t_0 < t$. For time $\tau \in [t_0, t)$ let the contents of the stack register* `stack` *be of the form*

$$\alpha_\tau = \langle \alpha_0, \ldots, \alpha_{k-1} \ , \rho(\tau), \ldots \rangle$$

with fixed $\alpha_0, \ldots, \alpha_{k-1}$ and variable $\rho(\tau) \leqslant \alpha_{k-1}$. Assume that the sequence $(\rho(\tau) | \tau \in [t_0, t))$ is weakly monotonously increasing and that the length of `stack` *is equal to $k+1$ cofinally often below t. Then at limit time t the content of* `stack` *is of the form*

$$\alpha_t = \langle \alpha_0, \ldots, \alpha_{k-1} \ , \rho \rangle$$

with $\rho = \bigcup_{\tau \in [t_0, t)} \rho(\tau)$.

5 A recursion theorem

Theorem 4 *Let $H : \mathrm{Ord}^3 \to \mathrm{Ord}$ be ordinal computable and define $F : \mathrm{Ord} \to \mathrm{Ord}$ recursively by*

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

Then F is ordinal computable.

Given an algorithm for the recursion function H we compute F with a `stack` as considered above and a register `value` which can hold a single value of the function F: we let `value = 2` stand for "undefined'. The following program P accepts an input ordinal α on the singleton stack $\langle\alpha\rangle$ and stops with the output stack $\langle\alpha\rangle$ and `value`$= F(\alpha)$. During the recursion the program will call itself with non-empty stacks $\alpha = \alpha_0, \alpha_1, \ldots, \alpha_{n-1}$ and compute the value $F(\alpha_{n-1})$. The main loop of the program serves to let the bounded quantifier $\exists\nu < \alpha$ range over all $\nu < \alpha$. The subloop evaluates the kernel $H(\alpha, \nu, F(\nu)) = 1$ of the quantifier and returns the result for further calculation of values.

```
  value:=2                %% set value to undefined
MainLoop:
  nu:=last(stack)
  alpha:=llast(stack)
  if nu = alpha then
1:  do
    remove_last_element_of(stack)
    value:=0              %% set value equal to 0
    goto SubLoop
    end
  else
2:  do
    stack:=stack + 1      %% push the ordinal 0 onto the stack
    goto MainLoop
    end
SubLoop:
  nu:=last(stack)
  alpha:=llast(stack)
  if alpha = UNDEFINED then STOP
  else
    do
    if H(alpha,nu,value)=1 then
3:    do
      remove_last_element_of(stack)
      value:=1
      goto SubLoop
      end
    else
4:    do
      stack:=stack + (3**y)*2  %% push y+1
      value:=2                 %% set value to undefined
      goto MainLoop
      end
    end
```

The correctness of the program is established by

Theorem 5 *The ordinal computation I, R by the program P has the following properties*

a) If I, R is in state `MainLoop` *at time s with* `stack` *contents $\langle\alpha_0, \ldots, \alpha_{n-1}\rangle$ where $n \geqslant 1$ then I, R will get into state* `SubLoop` *at a later time t with the same* `stack` *contents $\langle\alpha_0, \ldots, \alpha_{n-1}\rangle$ and the register* `value` *holding the value $F(\alpha_{n-1})$. Moreover in the interval $[s, t)$ the contents of* `stack` *will always be at least as big as $\langle\alpha_0, \ldots, \alpha_{n-1}\rangle$.*

b) Let I, R be in state `MainLoop` *at time s with* `stack` *contents $\alpha_0 > \ldots > \alpha_{n-1}$ where $n \geqslant 1$. Define $\bar{\alpha}$ = the minimal ordinal $\nu < \alpha_{n-1}$ such that $H(\alpha_{n-1}, \nu, F(\nu)) = 1$ if this exists and $\bar{\alpha} = \alpha_{n-1}$ else. Then there is a strictly increasing sequence $(t_i | i \leqslant \bar{\alpha})$ of times $t_i > t$ such that I, R is in state* `MainLoop` *at time t_i with* `stack` *contents $\langle\alpha_0, \ldots, \alpha_{n-1}, i\rangle$. Moreover in every time interval $[t_i, t_{i+1})$ the* `stack` *contents are $\geqslant \langle\alpha_0, \ldots, \alpha_{n-1}, i\rangle$.*

c) If I, R is in state `MainLoop` *with* `stack` *contents $\langle\alpha\rangle$ then it will later* stop *with* `stack` *contents $\langle\alpha\rangle$ and the register* `value` *holding the value $F(\alpha)$. Hence the function F is ordinal register computable.*

Proof a) and b) are proved simultaneously by induction over the last element α_{n-1} of the stack. Assume that P is in state `MainLoop` at time s with stack contents $\langle\alpha_0, \ldots, \alpha_{n-1}\rangle$ where $n \geqslant 1$ and that a) and b) hold for all stack contents $\langle\beta_0, \beta_1, \ldots, \beta_{m-1}\rangle$ with $\beta_{m-1} < \alpha_{n-1}$. Define $\bar{\alpha}$ as in b).

We first prove b) by defining an appropriate sequence $(t_i | i \leqslant \bar{\alpha})$ by recursion over $i \leqslant \bar{\alpha}$.
$i = 0$. Inspection of P shows that the computation will move to state 2 and obtain `stack` contents $\langle\alpha_0, \ldots, \alpha_{n-1}, 0\rangle$ before immediately returning to `MainLoop`.
$i = j+1$ where $j < \bar{\alpha}$. By recursion, P is in state `MainLoop` at time t_j with stack contents $\langle\alpha_0, \ldots, \alpha_{n-1}, j\rangle$. $j < \bar{\alpha} \leqslant \alpha_{n-1}$ so that the inductive assumption a) holds for $\langle\alpha_0, \ldots, \alpha_{n-1}, j\rangle$. So there will be a later time when P is in state `SubLoop` with stack contents $\langle\alpha_0, \ldots, \alpha_{n-1}, j\rangle$ and `value`$= F(j)$. Also during that computation the stack contents will always be $\geqslant \langle\alpha_0, \ldots, \alpha_{n-1}, j\rangle$. Inspection of the program shows that it will further compute $H(\alpha_{n-1}, j, F(j))$. This value will be $\neq 1$ by definition of $\bar{\alpha}$. So the computation will move on to state 4 with stack contents $\langle\alpha_0, \ldots, \alpha_{n-1}, j+1\rangle$. At the subsequent time $t_i = t_{j+1}$ the computation is in state `MainLoop` with stack contents $\langle\alpha_0, \ldots, \alpha_{n-1}, i\rangle$. i is a limit ordinal. Then by the limit behaviour of the machine and in particular by the above proposition, at time $t_i = \bigcup\{t_j | j < i\}$ the machine will be in state `MainLoop` with `stack` contents $\langle\alpha_0, \ldots, \alpha_{n-1}, i\rangle$.

Now we prove a).
Case 1: $\bar{\alpha} < \alpha_{n-1}$. Then $F(\bar{\alpha}) = 1$. By b) the computation will get to state `MainLoop` with `stack` contents $\langle\alpha_0, \ldots, \alpha_{n-1}, \bar{\alpha}\rangle$. By the inductive hypothesis, the machine will then get to state `SubLoop` with `stack` contents $\langle\alpha_0, \ldots, \alpha_{n-1}, \bar{\alpha}\rangle$ and `value` equal to $F(\bar{\alpha})$. Then the program will compute $H(\alpha_{n-1}, \bar{\alpha}, F(\bar{\alpha})) = 1$ and move into alternative 3. The register `value` obtains the value $F(\alpha_{n-1}) = 1$ and the computation moves to state `SubLoop` with the last stack element removed: `stack` $= \langle\alpha_0, \ldots, \alpha_{n-1}\rangle$, as required.

Case 2: $\bar{\alpha} = \alpha_{n-1}$. Then $F(\bar{\alpha}) = 0$. By b), the computation will get to state `MainLoop` with stack contents $\langle \alpha_0, \ldots, \alpha_{n-1}, \bar{\alpha} = \alpha_{n-1} \rangle$. Inspection of the program shows that it will get into alternative `1`, set `stack:=` $\langle \alpha_0, \ldots, \alpha_{n-1} \rangle$, `value:=` 0 and move to `SubLoop`, which proves a) in this case.

Finally, c) follows readily from a) and inspection of the program.

6 A recursive truth predicate

The ordinal arithmetic operations and the GÖDEL pairing function G allow us to code finite sequences of ordinals into single ordinals. The coding can be made ordinal computable in the sense that usual operations on finite sequences like concatenation or substitution are computable as well. This allows to code formal languages in an ordinal computable way.

We shall consider a language L_R appropriate for first-order structures of the type

$$(\alpha, <, G, R)$$

where the GÖDEL function G is viewed as a ternary *relation* on α and R is a unary relation on α. The terms of the language are variables v_n for $n < \omega$ and constant symbols c_ξ for $\xi \in \mathrm{Ord}$; the symbol c_ξ will be interpreted as the ordinal ξ. The language has atomic formulas $t_1 \equiv t_2$, $t_1 < t_2$, $\dot{G}(t_1, t_2, t_3)$ and $\dot{R}(t_1)$. The symbol $\dot{G}$ will be interpreted by the GÖDEL relation G. If φ and ψ are (compound) formulas of the language, $n < \omega$, and t is a term then

$$\neg\varphi, (\varphi \vee \psi), \text{ and } (\exists v_n < t)\ \varphi$$

are also formulas; thus we are only working with bounded quantifications. We assume an ordinal computable coding such that a bounded existential quantification $(\exists v_n < c_\xi)\ \varphi$ is coded by a larger ordinal than each of its instances $\varphi\frac{c_\zeta}{v_n}$ with $\zeta < \xi$:

$$\varphi\frac{c_\zeta}{v_n} < (\exists v_n < c_\xi)\ \varphi).$$

An L_R-formula is an L_R-*sentence* if it does not have free variables. If φ is an L_R-sentence so that all constants symbols c_ξ in φ have indices $\xi < \alpha$ then the satisfaction relation

$$(\alpha, <, G, R) \vDash \varphi$$

is defined as usual. Bounded sentences are *absolute* for sufficiently long initial segments of the ordinals. If φ is a bounded sentence such that every constant symbol c_ξ occuring in φ satisfies $\xi < \beta < \alpha$ then

$$(\alpha, <, G, R) \vDash \varphi \text{ iff } (\beta, <, G, R) \vDash \varphi.$$

We may assume that the coding of formulas by ordinals φ will satisfy that $\xi < \varphi$ for every constant symbol c_ξ occuring in φ. So the meaning of a bounded sentence φ is given by

$$(\varphi, <, G, R) \vDash \varphi.$$

This leads to the recursive definition of a *bounded truth predicate* $T \subseteq \mathrm{Ord}$ over the ordinals

$$T(\alpha) \text{ iff } \alpha \text{ is a bounded } L_R\text{-sentence and } (\alpha, <, G, T \cap \alpha) \vDash \alpha.$$

We shall see that T is a strong predicate which codes a model of set theory. We first show that the characteristic function χ_T of T can be defined according to the recursion scheme

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

with an appropriate computable recursion function H.

$$\begin{aligned} H(\alpha, \nu, \chi) = 1 \text{ iff } \alpha \text{ is an } L_R\text{-sentence and}& \\ &\exists \xi, \zeta < \alpha\ (\alpha = c_\xi \equiv c_\zeta \wedge \xi = \zeta) \\ &\text{or } \exists \xi, \zeta < \alpha\ (\alpha = c_\xi < c_\zeta \wedge \xi < \zeta) \\ &\text{or } \exists \xi, \zeta, \eta < \alpha\ (\alpha = \dot{G}(c_\xi, c_\zeta, c_\eta) \wedge \eta = G(\xi, \zeta)) \\ &\text{or } \exists \xi < \alpha\ (\alpha = \dot{R}(c_\xi) \wedge \nu = \xi \wedge \chi = 1) \\ &\text{or } \exists \varphi < \alpha\ (\alpha = \neg\varphi \wedge \nu = \varphi \wedge \chi = 0) \\ &\text{or } \exists \varphi, \psi < \alpha\ (\alpha = (\varphi \vee \psi) \wedge (\nu = \varphi \vee \nu = \psi) \wedge \chi = 1) \\ &\text{or } \exists n < \omega\ \exists \xi < \alpha\ \exists \varphi < \alpha \\ &\quad (\alpha = (\exists v_n < c_\xi)\ \varphi \wedge (\exists \zeta < \xi)\ \nu = \varphi \frac{c_\zeta}{v_n} \wedge \chi = 1). \end{aligned}$$

Then χ_T and T are ordinal register computable by the recursion theorem 4.

7 The theory SO of sets of ordinals

It is well-known that a model of Zermelo-Fraenkel set theory with the axiom of choice (ZFC) is determined by its sets of ordinals (see [6], Theorem 13.28). We define a natural theory SO which axiomatizes the sets of ordinals in a model of ZFC. The theory SO is two-sorted: *ordinals* are taken as atomic objects, the other sort corresponds to *sets of ordinals*. Let L_{SO} be the language

$$L_{\mathrm{SO}} := \{\mathrm{Ord}, \mathrm{SOrd}, <, =, \in, g\}$$

where Ord and SOrd are unary predicate symbols, $<$, $=$ and $\in$ are binary predicate symbols and g is a two-place function. To simplify notation, we use lower case greek letters to range over elements of Ord and lower case roman letters to range over elements of SOrd.

1. Well-ordering axiom:
$\forall \alpha, \beta, \gamma (\neg \alpha < \alpha \wedge (\alpha < \beta \wedge \beta < \gamma \to \alpha < \gamma) \wedge$
$(\alpha < \beta \vee \alpha = \beta \vee \beta < \alpha)) \wedge$
$\forall a (\exists \alpha (\alpha \in a) \to \exists \alpha (\alpha \in a \wedge \forall \beta (\beta < \alpha \to \neg \beta \in a)));$

2. Axiom of infinity (existence of a limit ordinal):
   $\exists\alpha(\exists\beta(\beta < \alpha) \wedge \forall\beta(\beta < \alpha \to \exists\gamma(\beta < \gamma \wedge \gamma < \alpha)))$;
3. Axiom of extensionality: $\forall a, b(\forall\alpha(\alpha \in a \leftrightarrow \alpha \in b) \to a = b)$;
4. Initial segment axiom: $\forall\alpha\exists a\forall\beta(\beta < \alpha \leftrightarrow \beta \in a)$;
5. Boundedness axiom: $\forall a\exists\alpha\forall\beta(\beta \in a \to \beta < \alpha)$;
6. Pairing axiom (GÖDEL Pairing Function):
   $\forall\alpha, \beta, \gamma(g(\beta, \gamma) \leq \alpha \leftrightarrow \forall\delta, \epsilon((\delta, \epsilon) <^* (\beta, \gamma) \to g(\delta, \epsilon) < \alpha))$.
   Here $(\alpha, \beta) <^* (\gamma, \delta)$ stands for
   $\exists\eta, \theta(\eta = \max(\alpha, \beta) \wedge \theta = \max(\gamma, \delta) \wedge (\eta < \theta \vee$
   $(\eta = \theta \wedge \alpha < \gamma) \vee (\eta = \theta \wedge \alpha = \gamma \wedge \beta < \delta)))$,
   where $\gamma = \max(\alpha, \beta)$ abbreviates $(\alpha > \beta \wedge \gamma = \alpha) \vee (\alpha \leq \beta \wedge \gamma = \beta)$;
7. g is onto: $\forall\alpha\exists\beta, \gamma\ (\alpha = g(\beta, \gamma))$;
8. Axiom schema of separation: For all L_{SO}-formulae $\phi(\alpha, P_1, \ldots, P_n)$ postulate:
   $\forall P_1, \ldots, P_n\forall a\exists b\forall\alpha(\alpha \in b \leftrightarrow \alpha \in a \wedge \phi(\alpha, P_1, \ldots, P_n))$;
9. Axiom schema of replacement: For all L_{SO}-formulae $\phi(\alpha, \beta, P_1, \ldots, P_n)$ postulate:
   $\forall P_1, \ldots, P_n(\forall\xi, \zeta_1, \zeta_2(\phi(\xi, \zeta_1, P_1, \ldots, P_n) \wedge \phi(\xi, \zeta_2, P_1, \ldots, P_n) \to \zeta_1 = \zeta_2)$
   $\to \forall a\exists b\forall\zeta(\zeta \in b \leftrightarrow \exists\xi \in a\ \phi(\xi, \zeta, P_1, \ldots, P_n)))$;
10. Powerset axiom:
   $\forall a\exists b\forall z(\exists\alpha(\alpha \in z) \wedge \forall\alpha(\alpha \in z \to \alpha \in a) \to \exists^{=1}\xi\forall\beta(\beta \in z \leftrightarrow g(\beta, \xi) \in b))$.

8 Assembling sets along wellfounded relations

In standard set theory a set x can be represented as a *point in a wellfounded relation*: consider the $\in$-relation on the transitive closure $\mathrm{TC}(\{x\})$ with distinguished element $x \in \mathrm{TC}(\{x\})$. By the MOSTOWSKI isomorphism theorem x is uniquely determined by the pair $(x, \mathrm{TC}(\{x\}))$ up to order isomorphism.

Definition 4 An ordered pair $x = (x, R_x)$ is a *point* if R_x is a wellfounded relation and $x \in \mathrm{dom}(R_x)$. Unless specified otherwise we use R_x to denote the wellfounded relation of the point x.

Obviously, $(x, \in\restriction \mathrm{TC}(\{x\}))$ is a point. Conversely, any point $x = (x, R_x)$ can be interpreted as a standard set $I(x)$. Define recursively

$$I_x : \mathrm{dom}(R_x) \to V,\ I_x(u) = \{I_x(v) | vRu\}.$$

Then let $I(x) = I_x(x)$ be the *interpretation* of x. Note that for points x and y

$$\begin{aligned} I_x(u) = I_y(v) \text{ iff } & \{I_x(u')|u'R_xu\} = \{I_x(v')|v'R_yv\} \\ \text{iff } & \forall u'R_xu\ \exists v'R_yv\ I_x(u') = I_y(v')) \wedge \\ & \wedge(\forall v'R_yv\ \exists u'R_xu\ I_x(u') = I_y(v'). \end{aligned}$$

This means that the relation $I_x(u) = I_y(v)$ in the variables u and v can be defined recursively without actually forming the interpretations $I_x(u)$ and

$I_y(v)$. Wellfounded relations and points can be handled within the theory SO. This will allow to define a model of ZFC within SO. So assume SO for the following construction.

Definition 5 Define a relation $\equiv$ on points $x = (x, R_x), y = (y, R_y)$ by induction on the product wellorder $R_x \times R_y$:

$$(x, R_x) \equiv (y, R_y) \text{ iff } \forall u R_x x \; \exists v R_y y \; (u, R_x) \equiv (v, R_y) \wedge$$
$$\wedge \forall v R_y y \; \exists u R_x x \; (u, R_x) \equiv (v, R_y).$$

Theorem 6 *$\equiv$ is an equivalence relation on points.*

Proof We only check transitivity; reflexivity and symmetry may be proved similarly.

Transitivity. Consider points $x = (x, R_x)$, $y = (y, R_y)$ and $z = (z, R_z)$. We show by induction on the wellfounded relation $R_x \times R_y \times R_z$ that

$$(u, R_x) \equiv (v, R_y) \wedge (v, R_y) \equiv (w, R_z) \to (u, R_x) \equiv (w, R_z).$$

Assume that the claim holds for all $u' R_x u$, $v' R_y v$ and $w' R_z w$. Assume that

$$(u, R_x) \equiv (v, R_y) \wedge (v, R_y) \equiv (w, R_z).$$

To show that $(u, R_x) \equiv (w, R_z)$ consider $u' R_x u$. By $(u, R_x) \equiv (v, R_y)$ take $v' R_y v$ such that $(u', R_x) \equiv (v', R_y)$. By $(v, R_y) \equiv (w, R_z)$ take $w' R_z w$ such that $(v', R_y) \equiv (w', R_z)$. By the inductive assumption, $(u', R_x) \equiv (v', R_y)$ and $(v', R_y) \equiv (w', R_z)$ imply that $(u', R_x) \equiv (w', R_z)$. Thus

$$\forall u' R_x u \; \exists w' R_z w \; (u', R_x) \equiv (w', R_z).$$

Similarly

$$\forall w' R_z w \; \exists u' R_x u \; (u', R_x) \equiv (w', R_z)$$

and thus $(u, R_x) \equiv (w, R_z)$. In particular for $x = (x, R_x)$, $y = (y, R_y)$ and $z = (z, R_z)$

$$x \equiv y \wedge y \equiv z \to x \equiv z.$$

We now define a membership relation for points.

Definition 6 Let $x = (x, R_x)$ and $y = (y, R_y)$ be points. Then set

$$x \blacktriangleleft y \text{ iff } \exists v R_y y \; x \equiv (v, R_y).$$

Lemma 1 *The equivalence relation $\equiv$ is a congruence relation with respect to $\blacktriangleleft$, i.e.,*

$$x \blacktriangleleft y \wedge x \equiv x' \wedge y \equiv y' \to x' \blacktriangleleft y'.$$

Proof Let $x \blacktriangleleft y \wedge x \equiv x' \wedge y \equiv y' \to x' \blacktriangleleft y'$. Take $v R_y y$ such that $x \equiv (v, R_y)$. By $y \equiv y'$ take $v' R_{y'} y'$ such that $v \equiv v'$. Since $\equiv$ is an equivalence relation, the relations $x \equiv x'$, $x \equiv v$ and $v \equiv v'$ imply $x' \equiv v'$. Hence $x' \blacktriangleleft y'$.

9 The class of points satisfies ZFC

We show that the class $\mathbb{P}$ of points with the relations $\equiv$ and $\blacktriangleleft$ satisfies the axioms ZFC of Zermelo-Fraenkel set theory with the axiom of choice. For the existence axioms of ZFC we prove a lemma about combining points into a single point.

Lemma 2 (SO) *Let $(x_i|i \in A)$ be a set-sized definable sequence of points, i.e., A is a set of ordinals and the function $i \mapsto x_i \in \mathbb{P}$ is definable. Then there is a point $y = (y, R_y)$ such that for all points x holds*

$$x \blacktriangleleft y \text{ iff } \exists i \in A\ x \equiv x_i.$$

Proof For $i \in A$ let $x_i = (x_i, R_i)$. Define points $x_i' = (x_i', R_i')$ by "colouring" every element of $\operatorname{dom}(R_i)$ by the "colour" i:

$$x_i' = (i, x_i) \text{ and } R_i' = \{((i,\alpha),(i,\beta))|(\alpha,\beta) \in R_i\}.$$

The points (x_i', R_i') and (x_i, R_i) are isomorphic and so $(x_i', R_i') \equiv (x_i, R_i)$. We may thus assume that the domains of the wellfounded relations R_i are pairwise disjoint. Take some $y \notin \bigcup_{i\in A} \operatorname{dom}(R_i)$ and define the point $y = (y, R_y)$ by

$$R_y = \bigcup_{i\in A} R_i \cup \{(x_i, y)|i \in A\}.$$

Consider $i \in A$. If $x \in \operatorname{dom}(R_i)$ then the iterated R_i-predecessors of x are equal to the iterated R_y-predecessors of x. Hence $(x, R_i) \equiv (x, R_y)$.

Assume now that $x \blacktriangleleft y$. Take $vR_y y$ such that $x \equiv (v, R_y)$. Take $i \in A$ such that $v = x_i$. By the previous remark

$$x \equiv (v, R_y) = (x_i, R_y) \equiv (x_i, R_i) = x_i.$$

Conversely consider $i \in A$ and $x \equiv x_i$. Then $x \equiv x_i = (x_i, R_i) \equiv (x_i, R_y)$ and $x_i R_y y$. This implies $x \blacktriangleleft y$.

We are now able to canonically interpret the theory ZFC within SO.

Theorem 7 (SO) $\mathbb{P} = (\mathbb{P}, \equiv, \blacktriangleleft)$ *is a model of* ZFC.

Proof (1) The axiom of *extensionality* holds in $\mathbb{P}$:

$$\forall x \forall y (\forall z (z \blacktriangleleft x \leftrightarrow z \blacktriangleleft y) \to x \equiv y).$$

Proof. Consider points x and y such that $\forall z (z \blacktriangleleft x \leftrightarrow z \blacktriangleleft y)$. Consider $uR_x x$. Then $(u, R_x) \blacktriangleleft (x, R_x) = x$. By assumption, $(u, R_x) \blacktriangleleft (y, R_y)$. By definition take $vR_y y$ such that $(u, R_x) \equiv (v, R_y)$. Thus

$$\forall u R_x x\ \exists v R_y y\ (u, R_x) \equiv (v, R_y).$$

By exchanging x and y one also gets

$$\forall v R_y y\ \exists u R_x x\ (u, R_x) \equiv (v, R_y).$$

Hence $x \equiv y$. *qed*(1)

(2) The axiom of *pairing* holds in $\mathbb{P}$:

$$\forall x \forall y \exists z \forall w (w \blacktriangleleft z \leftrightarrow (w \equiv x \vee w \equiv y)).$$

Proof. Consider points $x = (x, R_x)$ and $y = (y, R_y)$. By the comprehension lemma 2 there is a point $z = (z, R_z)$ such that for all points w

$$w \blacktriangleleft z \leftrightarrow (w \equiv x \vee w \equiv y).$$

qed(2)

(3) The axiom of *unions* holds in $\mathbb{P}$:

$$\forall x \exists y \forall z (z \blacktriangleleft y \leftrightarrow \exists w (w \blacktriangleleft x \wedge z \blacktriangleleft w)).$$

Proof. Consider a point $x = (x, R_x)$. Let

$$A = \{i \in \mathrm{dom}(R_x) | \exists u \in \mathrm{dom}(R_x)\ iR_x u R_x x\}.$$

For $i \in A$ define the point $x_i = (i, R_x)$. By the Comprehension Lemma 2 there is a point $y = (y, R_y)$ such that for all points z

$$z \blacktriangleleft y \leftrightarrow \exists i \in A\ z \equiv x_i.$$

To show the axiom consider some $z \blacktriangleleft y$. Take $i \in A$ such that $z \equiv x_i$. Take $u \in \mathrm{dom}(R_x)$ such that $iR_x u R_x x$. Then $z \equiv x_i = (i, R_x) \blacktriangleleft (u, R_x) \blacktriangleleft (x, R_x) = x$, i.e., $\exists w (z \blacktriangleleft w \blacktriangleleft x)$.

Conversely assume that $\exists w (z \blacktriangleleft w \blacktriangleleft x)$ and take w such that $z \blacktriangleleft w \blacktriangleleft x$. Take $uR_x x$ such that $w \equiv (u, R_x)$. Then $z \blacktriangleleft (u, R_x)$. Take $iR_x u$ such that $z \equiv (i, R_x) = x_i$. Then $z \blacktriangleleft y$. *qed*(3)

(4) The *replacement schema* holds in $\mathbb{P}$, i.e., for every first-order formula $\varphi(u, v)$ in the language of $\equiv$ and $\blacktriangleleft$ the following is true in $\mathbb{P}$:

$$\forall u, v, v' ((\varphi(u,v) \wedge \varphi(u,v')) \rightarrow v \equiv v') \rightarrow \forall x \exists y \forall z (z \blacktriangleleft y \leftrightarrow \exists u (u \blacktriangleleft x \wedge \varphi(u, z))).$$

Proof. Note that the formula φ may contain further free parameters, which we do not mention for the sake of simplicity. Assume that $\forall u, v, v' ((\varphi(u,v) \wedge \varphi(u,v')) \rightarrow v \equiv v')$ and let $x = (x, R_x)$ be a point. Let $A = \{i | iR_x x\}$. For each $i \in A$ we have the point $(i, R_x) \blacktriangleleft (x, R_x) = x$. Using replacement and choice in SO we can pick for each $i \in A$ a point $z_i = (z_i, R_{z_i})$ such that $\varphi((i, R_x), z_i)$ holds if such a point exists. By the Comprehension Lemma 2 there is a point $y = (y, R_y)$ such that for all points z

$$z \blacktriangleleft y \leftrightarrow \exists i \in A\ z \equiv z_i.$$

To show the instance of the replacement schema under consideration, assume that $z \blacktriangleleft y$. Take $i \in A$ such that $z \equiv z_i$. Then $(i, R_x) \blacktriangleleft (x, R_x) = x$, $\varphi((i, R_x), z_i)$ and $\varphi((i, R_x), z)$. Hence $\exists u (u \blacktriangleleft x \wedge \varphi(u, z))$.

Conversely, assume that $\exists u (u \blacktriangleleft x \wedge \varphi(u, z))$. Take $u \blacktriangleleft x$ such that $\varphi(u, z)$. Take $iR_x x$, $i \in A$ such that $u \equiv (i, R_x)$. Then $\varphi((i, R_x), z)$. By definition of

z_i, $\varphi((i,R_x),z_i)$. The functionality of the formula φ implies $z \equiv z_i$. Hence $\exists i \in A\ z \equiv z_i$ and $z \blacktriangleleft y$. *qed*(4)

The replacement schema also implies the *separation* schema.

(5) The axiom of *powersets* holds in $\mathbb{P}$:

$$\forall x \exists y \forall z (z \blacktriangleleft y \leftrightarrow \forall w (w \blacktriangleleft z \rightarrow w \blacktriangleleft x)).$$

Proof. By the separation schema it suffices to show that

$$\forall x \exists y \forall c (\forall w (w \blacktriangleleft c \rightarrow w \blacktriangleleft x) \rightarrow c \blacktriangleleft y).$$

Consider a point $x = (x, R_x)$. Let $F = \mathrm{dom}(R_x) \cup \mathrm{ran}(R_x)$ be the *field* of R_x. By the powerset axiom of SO choose some set P such that $\mathrm{Pow}(P,F)$:

$$\forall z (\exists \alpha (\alpha \in z) \wedge \forall \alpha (\alpha \in z \rightarrow \alpha \in F) \rightarrow \exists \xi \forall \beta (\beta \in z \leftrightarrow (\beta, \xi) \in P)).$$

Choose two large ordinals δ and y such that

$$\forall \alpha \in F\ \alpha < \delta \text{ and } \forall \xi (\xi \in \mathrm{ran}(P) \rightarrow (\delta, \xi) < y).$$

Define a point $y = (y, R_y)$ by

$$R_y = R_x \cup \{(\beta, (\delta, \xi)) | (\beta, \xi) \in P\} \cup \{((\delta, \xi), y) | \xi \in \mathrm{ran}(P)\}.$$

To show the axiom consider some point $c = (c, R_c)$ such that $\forall w (w \blacktriangleleft c \rightarrow w \blacktriangleleft x)$. Define a corresponding subset z of F by

$$z = \{\beta \in F | \exists v R_c c\ (v, R_c) \equiv (\beta, R_x)\}.$$

We may assume for simplicity that $z \neq \emptyset$. By the powerset axiom of SO choose $\xi \in \mathrm{ran}(P)$ such that

$$\forall \beta (\beta \in z \leftrightarrow (\beta, \xi) \in P).$$

We claim that $((\delta, \xi), R_y) \equiv c$ and thus $c \blacktriangleleft y$.

Consider $\beta R_y (\delta, \xi)$. By the definition of $R_y$we have $(\beta, \xi) \in P$ and so $\beta \in z$. By the definition of z choose $v R_c c$ such that $(v, R_c) \equiv (\beta, R_x) \equiv (\beta, R_y)$.

Conversely, consider $v R_c c$. Then $(v, R_c) \blacktriangleleft (c, R_c) = c$. The subset property implies $(v, R_c) \blacktriangleleft (x, R_x) = x$. Take $\beta R_x x$ such that $(v, R_c) \equiv (\beta, R_x) \equiv (\beta, R_y)$. By definition, $\beta \in z$, $(\beta, \xi) \in P$ and $\beta R_y (\delta, x)$. *qed*(5)

(6) The *axiom of choice* holds in $\mathbb{P}$:

$$\forall x ((\forall y, z (y \blacktriangleleft x \wedge z \blacktriangleleft x \rightarrow (\exists u\ u \blacktriangleleft y \wedge (\neg y \equiv z \rightarrow \neg \exists u (u \blacktriangleleft y \wedge u \blacktriangleleft z))))) \rightarrow$$
$$\rightarrow \exists w \forall y (y \blacktriangleleft x \rightarrow \exists u ((u \blacktriangleleft w \wedge u \blacktriangleleft y) \wedge \forall v ((v \blacktriangleleft w \wedge v \blacktriangleleft y) \rightarrow u \equiv v)))).$$

Proof. Let $x = (x, R_x) \in \mathbb{P}$ be a point such that

$$\forall y, z ((y \blacktriangleleft x \wedge z \blacktriangleleft x) \rightarrow (\exists u\ u \blacktriangleleft y \wedge (\neg y \equiv z \rightarrow \neg \exists u (u \blacktriangleleft y \wedge u \blacktriangleleft z)))).$$

Choose an ordinal $\alpha \in \mathrm{dom}(R_x)$ and define the point $w = (\alpha, R_w)$ by letting its “elements” be least ordinals in the “elements” of x:

$$R_w = R_x \cup \{(\xi, \alpha) | \exists \zeta (\xi R_x \zeta R_x x \wedge$$
$$\wedge (\forall \xi' < \xi\ \forall \zeta' ((\zeta, R_x) \equiv (\zeta', R_x) \rightarrow \neg (\xi R_x \xi' R_x \zeta))))\}.$$

To show that w witnesses the axiom of choice for x consider a point y with $y \blacktriangleleft x$. We may assume that y is of the form $y = (\zeta, R_x)$ where $\zeta R_x x$. By the assumption on x there exists $u \blacktriangleleft y$. Take some ξ such that $(\xi, R_x) \equiv u$. We may assume that ζ and ξ with these properties are chosen so that ξ is minimal in the ordinals. Then

$$\xi R_x \zeta R_x x \wedge (\forall \xi' < \xi \; \forall \zeta' ((\zeta, R_x) \equiv (\zeta', R_x) \rightarrow \neg(\xi R_x \xi' R_x \zeta))) \qquad (1)$$

and so $\xi R_w \alpha$. Thus $u \blacktriangleleft w$. To show the uniqueness of this u with $u \blacktriangleleft w \wedge u \blacktriangleleft y$ consider some v with $v \blacktriangleleft w \wedge v \blacktriangleleft y$. We may assume that v is of the form $v = (\xi', R_w)$ with $\xi' R_w \alpha$. By the definition of R_w we choose some ζ' such that

$$\xi' R_x \zeta' R_x x \wedge (\forall \xi'' < \xi' \; \forall \zeta'' ((\zeta', R_x) \equiv (\zeta'', R_x) \rightarrow \neg(\xi' R_x \xi'' R_x \zeta'))). \qquad (2)$$

Now

$$v \blacktriangleleft y \blacktriangleleft x \text{ and } v = (\xi', R_w) \blacktriangleleft (\zeta', R_w) \blacktriangleleft (x, R_x) = x.$$

Since the "elements" of x are "pairwise disjoint", we have $y \equiv (\zeta', R_w)$. Since $y \equiv (\zeta, R_x)$ the conditions (2) and (3) become equivalent and define the same ordinal $\xi = \xi'$. Hence

$$u \equiv (\xi, R_x) \equiv (\xi', R_w) \equiv v.$$

qed(6)

(7) The *foundation schema* holds in $\mathbb{P}$, i.e., for every first-order formula $\varphi(u)$ in the language of $\equiv$ and $\blacktriangleleft$ the following is true in $\mathbb{P}$:

$$\exists u \; \varphi(u) \rightarrow \exists y (\varphi(y) \wedge \forall z (z \blacktriangleleft y \rightarrow \neg \varphi(z))).$$

Proof. Note that the formula φ may contain further free parameters, which we do not mention for the sake of simplicity. Assume that $\exists u \; \varphi(u)$. Take a point $x = (x, R_x)$ such that $\varphi(x)$. Since R_x is wellfounded one may take an R_x-minimal $y \in \mathrm{dom}(R_x)$ such that $\varphi((y, R_x))$. Letting y also denote the point (y, R_x) then $\varphi(y)$. To prove the axiom, consider some point $z \blacktriangleleft y$. Take $v R_x y$ such that $z \equiv (v, R_x)$. By the R_x-minimal choice of y we have $\neg\varphi((v, R_x))$. Hence $\neg\varphi(z)$. *qed*(7)

(8) The axiom of *infinity* holds in $\mathbb{P}$, i.e.,

$$\exists x ((\exists y \; y \blacktriangleleft x) \wedge (\forall y (y \blacktriangleleft x \rightarrow \exists z (z \blacktriangleleft x \wedge \forall u (u \blacktriangleleft z \leftrightarrow (u \blacktriangleleft y \vee u \equiv y))))))$$

Proof. In SO let ω be the smallest limit ordinal. We show that

$$x = (\omega, < \restriction (\omega + 1)^2)$$

witnesses the axiom. Since $(0, < \restriction (\omega+1)^2) \blacktriangleleft (\omega, < \restriction (\omega+1)^2)$ we have $\exists y \; y \blacktriangleleft x$. Consider some $y \blacktriangleleft x$. We may assume that $y = (n, < \restriction (\omega+1)^2)$ for some $n < \omega$. Set

$$z = (n + 1, < \restriction (\omega + 1)^2).$$

It is easy to check that

$$z \blacktriangleleft x \wedge \forall u (u \blacktriangleleft z \leftrightarrow (u \blacktriangleleft y \vee u \equiv y)).$$

qed(8)

10 T codes a model of SO

The truth predicate T contains information about a large class of sets of ordinals.

Definition 7 For ordinals μ and α define

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

Set

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

Theorem 8 $(\mathrm{Ord},\mathcal{S},<,=,\in,G)$ *is a model of the theory* SO.

Proof The axioms (1)-(7) are obvious. The proofs of axiom schemas (8) and (9) rest on a Levy-type reflection principle. For $\theta\in\mathrm{Ord}$ define

$$\mathcal{S}_\theta=\{X(\mu,\alpha)|\mu,\alpha\in\theta\}.$$

Then for any L_{SO}-formula $\varphi(v_0,\dots,v_{n-1})$ and $\eta\in\mathrm{Ord}$ there is some limit ordinal $\theta>\eta$ such that

$$\forall\xi_0,\dots,\xi_{n-1}\in\theta((\mathrm{Ord},\mathcal{S},<,=,\in,G)\models\varphi[\xi_0,\dots,\xi_{n-1}]\text{ iff}$$
$$\text{iff }(\theta,\mathcal{S}_\theta,<,=,\in,G)\models\varphi[\xi_0,\dots,\xi_{n-1}]).$$

Since all elements of $\mathcal{S}_\theta$ can be defined from the truth function T and ordinals $<\theta$, the right-hand side can be evaluated in the structure $(\theta,<,G\cap\theta^3,T)$ by an L_R-formula φ^* which can be recursively computed from φ. Hence

$$\forall\xi_0,\dots,\xi_{n-1}\in\theta((\mathrm{Ord},\mathcal{S},<,=,\in,G)\models\varphi[\xi_0,\dots,\xi_{n-1}]\text{ iff}$$
$$\text{iff }(\theta,<,G\cap\theta^3,T)\models\varphi^*[\xi_0,\dots,\xi_{n-1}]).$$

So sets witnessing axioms (8) and (9) can be defined over $(\theta,<,G\cap\theta^3,T)$ and are thus elements of $\mathcal{S}$.

The powerset axiom of SO can be shown by a similar reflection argument.

11 Ordinal computability corresponds to constructibility

Kurt Gödel [4] defined the inner model L of *constructible sets* as the union of a hierarchy of levels L_α:

$$L=\bigcup_{\alpha\in\mathrm{Ord}}L_\alpha$$

where the hierarchy is defined by: $L_0=\emptyset$, $L_\delta=\bigcup_{\alpha<\delta}L_\alpha$ for limit ordinals δ, and $L_{\alpha+1}=$ the set of all sets which are first-order definable in the structure $(L_\alpha,\in)$. The model L is the $\subseteq$-smallest inner model of set theory. The standard reference for the theory of the model L is the monograph [3].

The following main result provides a characterization of ordinal register computability which does not depend on a specific machine model or coding of language:

Theorem 9 *A set x of ordinals is ordinal computable if and only if it is an element of the constructible universe L.*

Proof Let $x \subseteq \mathrm{Ord}$ be ordinal computable by the program P from the ordinals $\delta_1, \ldots, \delta_{n-1}$, so that for every $\alpha \in \mathrm{Ord}$:

$$P : (\alpha, \delta_1, \ldots, \delta_{n-1}, 0, 0, \ldots) \mapsto \chi_x(\alpha).$$

By the simple nature of the computation procedure the same computation can be carried out inside the inner model L, so that for every $\alpha \in \mathrm{Ord}$:

$$(L, \in) \models P : (\alpha, \delta_1, \ldots, \delta_{n-1}, 0, 0, \ldots) \mapsto \chi_x(\alpha).$$

Hence $\chi_x \in L$ and $x \in L$.

Conversely consider $x \in L$. Since $(\mathrm{Ord}, \mathcal{S}, <, =, \in, G)$ is a model of the theory SO there is an inner model M of set theory such that

$$\mathcal{S} = \{z \subseteq \mathrm{Ord} \,|\, z \in M\}.$$

Since L is the $\subseteq$-smallest inner model, $L \subseteq M$. Hence $x \in M$ and $x \in \mathcal{S}$. Let $x = X(\mu, \alpha)$. By the computability of the truth predicate, x is ordinal register computable from the parameters μ and α.

References

[1] Ryan Bissell-Siders. Ordinal computers. Eprint at: arXiv:math.LO/9804076, 1998.
[2] Nigel J. Cutland. *Computability: An introduction to Recursive Function Theory.* Perspectives in Mathematical Logic. Cambridge University Press, 1980.
[3] Keith Devlin. *Constructibility.* Perspectives in Mathematical Logic. Springer-Verlag, Berlin, 1984.
[4] Kurt Gödel. *The Consistency of the Continuum Hypothesis*, volume 3 of *Ann. of Math. Studies.* Princeton University Press, Princeton, 1940.
[5] Joel David Hamkins and Andy Lewis. Infinite Time Turing Machines. *J. Symbolic Logic*, 65(2):567–604, 2000.
[6] Thomas Jech. *Set Theory. The Third Millennium Edition.* Springer Monographs in Mathematics. Springer-Verlag, 2003.
[7] Peter Koepke. Turing computations on ordinals. *Bull. Symbolic Logic*, 11(3):377–397, 2005.