Fundamenta Informaticae 34 (2001) 1–9
IOS Press

Computability in type-2 objects with well-behaved type-1 oracles is p-normal

George Tourlakis
Department of Computer Science
York University
Toronto, Canada
gt@cs.yorku.ca

Abstract. We show that computability in a type-2 object is p-normal if type-1 *partial* inputs are computed by "well-behaved oracles".

Keywords: Computability, type-2 computability, oracles, p-normality.

1. Introduction

In [6, 7] we introduced and studied a formalism for the computability of (type-2) functionals that allow *partial* type-1 inputs. A central feature of the formalism was the presence of a "clock", postulated by the inclusion of "computations" $\{a\}(t, x, \alpha) = z$ ($z \in \{0, 1\}$) in the standard Kleene-schemata list, so that $z = 0$ iff the "program" a on (partial) type-1 input α will receive an answer for the "oracle"-computation $\alpha(x)$ within t "steps".

The type-1 oracles allowed were "well-behaved" in two respects: If a query "$\{e\}(x) = ?$" was presented to them, then they used the program e to compute the answer. If, however, we asked "$\alpha(x) = ?$", in ignorance of a program for α, then the oracle would use its own "secret algorithm" to compute the result, being *deterministic* about it in the sense that its behaviour for any given query would be always the same. It was shown in [6] that the set of Moschovakis' *search-computable* single-valued functionals ([5]) is properly contained in the new theory—the reason being, partly, that search-computable functionals are consistent while the new computability can compute *inconsistent* (or, *non monotone*) functionals.[1]

[1] An important example of an "intuitively computable" inconsistent functional, when partial type-1 inputs are allowed, is the H of Theorem 2.2 below.

In this paper we extend our computable functionals by one type up, allowing also type-2 inputs, or, more conveniently, doing our computations *relative to* (or, *in*) *a fixed type-*2 *functional*, which we will generically call "$\mathbb{I}$". Our main result (Theorem 2.2) is that the clock-axiom "helps" this higher type computability to be p-normal.[2] That is, the functional that compares the lengths of two computations—i.e., computation tree depths (see Definition 2.4)—is formally computable.

As it is customary in the literature, $\mathbb{I} = \lambda\alpha.\mathbb{I}(\alpha)$, i.e., $\mathbb{I}$ will have just one argument, the (type-1) object α. Moreover, it will be convenient to assume that $\mathbb{I}$ is a *total restricted* functional, where "restricted" means that it is undefined on all *non total* α, and "total" means that it is defined on *all* total α.

A computation $F(\vec{x}, \vec{\alpha})$ relative to a fixed type-2 functional $\mathbb{I}$—where $\vec{x} = x_1, \ldots, x_n$ is a sequence of n inputs from the natural numbers (we write $\vec{x}_n$ if we must refer to n) while $\vec{\alpha} = \alpha_1, \ldots, \alpha_l$ is a sequence of l type-1 inputs[3]—proceeds as usual, "calling" the oracle for α_j whenever the value $\alpha_j(y)$ is needed. During the computation it may also be that the value $\mathbb{I}(\lambda y.G(y, \vec{x}, \vec{\alpha}))$ is needed, where G is given by a program e ($G = \{e\}$). A (*type*-2) oracle for $\mathbb{I}$ will effect this sub-computation and pass an answer (informed by $\vec{x}, \vec{\alpha}$ and e) *once it is satisfied that* $(\forall y)\{e\}(y, \vec{x}, \vec{\alpha}) \downarrow$.[4]

2. $\Pi_{\mathbb{I}}$-computability relative to a total type-2 functional $\mathbb{I}$

The following definition of the theory $\Pi_{\mathbb{I}}$ uses Kleene-schemata ([3]). I–X are "standard", while XI introduces a "clock" for type-1 oracle (finite) computations ([6, 7]) with the *intended semantics* given below.

$$\text{For all } t, x, \alpha,\ \mathbf{X}(t, x, \alpha) = \textbf{ if } \alpha(x)\downarrow \text{ in } \le t \text{ steps } \textbf{then } 0 \textbf{ else } 1$$

Technically, we add to the set of "initial functionals" a *total* functional $\mathbf{X}$ that satisfies:

(i) The range of $\mathbf{X}$ is a subset of $\{0, 1\}$,

(ii) for any x, α,
$$\alpha(x)\downarrow \text{ iff } (\exists t \in \omega)\mathbf{X}(t, x, \alpha) = 0$$

(iii) for all t, x, α, if $\mathbf{X}(t, x, \alpha) = 0$, then also $\mathbf{X}(t + 1, x, \alpha) = 0$.

Condition (ii) above captures our (semantical) intention that the "hidden algorithm" that a type-1 oracle uses to compute $\alpha(x)$ is oblivious to the presence or absence of type-2 oracles, and therefore t is still a finite ordinal (if $\alpha(x)\downarrow$) as it naturally is in the unrelativized theory. The reader will note that adding the initial functional $\mathbf{X}$ is analogous to the standard practice of adding the "evaluation functional" $\mathbf{Ev}$ that is given for all x, α by $\mathbf{Ev}(x, \alpha) = \alpha(x)$. However,

[2] The "help" manifests itself in the proof of Theorem 2.2.

[3] We say that F has *rank* (n, l).

[4] $f(a)\downarrow$ means that $f(a)$ is defined, while $f(a)\uparrow$ means that $f(a)$ is undefined. These infinitely many sub-computations done by the type-2 oracle, one for each $y \in \omega$, are required because $\mathbb{I}$ is defined on total inputs only. The oracle checks for input validity.

whereas the latter is uniquely determined by the *extension* of α—i.e., the set of tuples $\langle x, y\rangle$ that belong to α—the choice of $\mathbf{X}$ depends on the *intention* of the oracle for α, but this is "unknown". Technically, there are infinitely many ways to choose $\mathbf{X}$ subject to (i)–(iii) above, but we are not ready to suggest criteria that will allow one to prefer one clock $\mathbf{X}$ over another for being more "natural".

The "standard" clause XII is added to I–XI of [6, 7] and introduces the type-2 oracle which "computes" the fixed functional $\mathbb{I}$. For technical convenience we have followed the "custom" of restricting attention to a *total restricted* $\mathbb{I}$.

Definition 2.1. Let $k = \text{length}(\vec{x})$ and $l = \text{length}(\vec{\alpha})$. The set $\Pi_{\mathbb{I}}$ of *computations relative to* $\mathbb{I}$ is the *smallest* set of tuples $(e, \vec{x}, \vec{\alpha}, y)$ satisfying I–XI below:[5]

I. $(\langle 0, k, l, i\rangle, \vec{x}, \vec{\alpha}, x_i) \in \Pi_{\mathbb{I}}$ **for** $1 \le i \le k$

II. $(\langle 1, k, l, i\rangle, \vec{x}, \vec{\alpha}, x_i + 1) \in \Pi_{\mathbb{I}}$ **for** $1 \le i \le k$

III. $(\langle 2, k, l, c\rangle, \vec{x}, \vec{\alpha}, c) \in \Pi_{\mathbb{I}}$ **for** $c \in \omega$

IV. $(\langle 3, k, l\rangle, \vec{x}, \vec{\alpha}, \langle\vec{x}\rangle) \in \Pi_{\mathbb{I}}$

V. $(\langle 4, k+4, l\rangle, z, y, u, v, \vec{x}, \vec{\alpha}, z) \in \Pi_{\mathbb{I}}$ **if** $u = v$

$(\langle 4, k+4, l\rangle, z, y, u, v, \vec{x}, \vec{\alpha}, y) \in \Pi_{\mathbb{I}}$ **if** $u \ne v$

For any vector $\vec{c}$ of distinct numbers

VI. $(\langle 5, k+r+1, l, r, \vec{c}\rangle, y, \vec{z}, \vec{x}, \vec{\alpha}, z_i) \in \Pi_{\mathbb{I}}$ **for** $y = c_i,\ i = 1, \ldots, r$

where $\text{length}(\vec{c}) = \text{length}(\vec{z}) = r$

VII. $(\langle 6, k, l, i, j\rangle, \vec{x}, \vec{\alpha}, y) \in \Pi_{\mathbb{I}}$ **if** $\alpha_j(x_i) = y$

VIII. $(\langle 7, k+m+1, l, m\rangle, f, \vec{e}_m, \vec{x}, \vec{\alpha}, y) \in \Pi_{\mathbb{I}}$ **if** $(f, \vec{z}_m, \vec{x}, \vec{\alpha}, y) \in \Pi_{\mathbb{I}}$

and $(e_i, \vec{x}, \vec{\alpha}, z_i) \in \Pi_{\mathbb{I}}$ **for** $i = 1, \ldots, m$

IX. $(\langle 8, k, l, m, e, \vec{y}_m\rangle, \vec{x}, \vec{\alpha}, z) \in \Pi_{\mathbb{I}}$ **if** $(e, \vec{y}_m, \vec{x}, \vec{\alpha}, z) \in \Pi_{\mathbb{I}}$

X. $(\langle 9, k+1, l\rangle, e, \vec{x}, \vec{\alpha}, y) \in \Pi_{\mathbb{I}}$ **if** $(e, \vec{x}, \vec{\alpha}, y) \in \Pi_{\mathbb{I}}$

The "clock" axiom

XI. $(\langle 10, k+1, l, i, j\rangle, y, \vec{x}, \vec{\alpha}, z) \in \Pi_{\mathbb{I}}$ **if** $\mathbf{X}(y, x_i, \alpha_j) = z$

The $\mathbb{I}$ axiom

XII. $(\langle 11, k, l, e\rangle, \vec{x}, \vec{\alpha}, y) \in \Pi_{\mathbb{I}}$ **if** $(\forall z)(\exists w)(e, z, \vec{x}, \vec{\alpha}, w) \in \Pi_{\mathbb{I}}$

and $\mathbb{I}(\lambda z.\{e\}(z, \vec{x}, \vec{\alpha})) = y$

Intuitively, the y component in $(\langle 10, k+1, l, i, j\rangle, y, \vec{x}, \vec{\alpha}, z)$ is the "number of steps" registered in the clock—at some point in time—for the oracle's computation of $\alpha_j(x_i)$. If $z = 0$ then the computation actually terminated in y steps. If $z = 1$ then the oracle is still computing.

The oracle for $\mathbb{I}$ "checks" that $\lambda z.\{e\}(z, \vec{x}, \vec{\alpha})$ is total (the if-part in clause XII) and, if so, it computes the answer y which depends on $\vec{x}$ and $\vec{\alpha}$.

[5] $(\ldots)$ denotes (set-theoretic) ordered tuples, while $\langle\cdots\rangle$ denotes the usual coding: For the empty sequence Λ we set $\langle\Lambda\rangle = 1$. Moreover, $\langle x_0, \ldots x_{n-1}\rangle = \prod_{i=0}^{n-1} p_i^{x_i+1}$, where p_i is the i-th prime ($p_0 = 2$).

We have included clause VI for technical convenience, so that "table look-up" involved in a definition such as

$$f(y,\vec{z}) = \begin{cases} z_1 & \textbf{if } y = c_1 \textbf{ else} \\ \vdots & \vdots \\ z_r & \textbf{if } y = c_r \textbf{ else} \\ \uparrow & \end{cases} \tag{1}$$

is as "easy" to compute as it is *intuitively expected* to be. If (1) were to be simulated by clause V and composition (clause VIII), then the computation depth[6] for an input value c_i (read into the variable y) would depend not on any intrinsic properties of the input (e.g. input size), but instead on the position of the test "$y = c_i$" in the table (due to the nesting of the **if-then-else** clause V).

$\{e\}^{\Pi}_{\mathbb{I}}(\vec{x},\vec{\alpha}) = y$ means $(e,\vec{x},\vec{\alpha},y) \in \Pi_{\mathbb{I}}$. It is trivial to verify that the set of computation tuples $\Pi_{\mathbb{I}}$ is single-valued in the rightmost argument, therefore the functionals $\lambda\vec{x}\vec{\alpha}.\{e\}^{\Pi}_{\mathbb{I}}(\vec{x},\vec{\alpha})$ are single-valued. We drop the superscript Π from $\{e\}^{\Pi}_{\mathbb{I}}$ from now on, however the subscript $\mathbb{I}$ will be explicit. Thus $\{a\}_{\mathbb{I}}$ is computed according to the clauses I–XII, while $\{a\}$ is computed according to clauses I–XI.

Definition 2.2. The set of *partial* $\Pi_{\mathbb{I}}$-*computable* functionals, $\big\{\{e\}_{\mathbb{I}} : e \in \omega\big\}$, is denoted by $\mathcal{P}^{\Pi}_{\mathbb{I}}$. Thus, $F \in \mathcal{P}^{\Pi}_{\mathbb{I}}$ iff $F = \{e\}_{\mathbb{I}}$ for some $e \in \omega$. The set of $\Pi_{\mathbb{I}}$-*computable* functionals, $\mathcal{R}^{\Pi}_{\mathbb{I}}$, is the set of *total* functionals in $\mathcal{P}^{\Pi}_{\mathbb{I}}$. By dropping clause XII we go back to the *unrelativized* sets $\mathcal{P}^{\Pi}$ and $\mathcal{R}^{\Pi}$ of [6, 7]. The terms "computable" and "recursive" are synonymous.

The next two definitions define *immediate subcomputations*, i.s., and computation(-tree) *depths*. Since computations $(e,\vec{x},\vec{\alpha},y)$ are single-valued in y they can be unambiguously denoted by their "truncated" counterparts $(e,\vec{x},\vec{\alpha})$.

Definition 2.3. (a) I–VI have no i.s.

(b) $(\langle 10, k+1, l, i, j\rangle, y, \vec{x}, \vec{\alpha}, 0)$ has no i.s.

(c) $(\langle 6, k, l, i, j\rangle, \vec{x}, \vec{\alpha})$ has $(\langle 10, k+1, l, i, j\rangle, 0, \vec{x}, \vec{\alpha})$ as its only i.s.[7]

(d) $(\langle 10, k+1, l, i, j\rangle, y, \vec{x}, \vec{\alpha}, 1)$ has $(\langle 10, k+1, l, i, j\rangle, y+1, \vec{x}, \vec{\alpha})$ as its only i.s.

(e) The only i.s. of $(\langle 7, k+m+1, l, m\rangle, f, \vec{e}_m, \vec{x}, \vec{\alpha})$ are $(e_i, \vec{x}, \vec{\alpha})$ for $i = 1, \ldots, m$, and $(f, \{e_1\}(\vec{x},\vec{\alpha}), \ldots, \{e_m\}(\vec{x},\vec{\alpha}), \vec{x}, \vec{\alpha})$.

(f) $(\langle 8, k, l, m, e, \vec{y}_m\rangle, \vec{x}, \vec{\alpha})$ has $(e, \vec{y}_m, \vec{x}, \vec{\alpha})$ as its only i.s.

(g) The only i.s. of $(\langle 9, k+1, l\rangle, e, \vec{x}, \vec{\alpha})$ is $(e, \vec{x}, \vec{\alpha})$.

(h) $(\langle 11, k, l, e\rangle, \vec{x}, \vec{\alpha})$ has $(e, y, \vec{x}, \vec{\alpha})$, for all $y \in \omega$, as its i.s.

The *subcomputation* relation is the transitive closure of i.s.

Definition 2.4. If $u = (e,\vec{x},\vec{\alpha})$ is a computation, then its *depth*, $\|u\|$, is an ordinal defined as follows: If u falls under clauses I–VI, or if $u = (\langle 10, k+1, l, i, j\rangle, y, \vec{x}, \vec{\alpha}, 0) \in \Pi_{\mathbb{I}}$, then $\|u\| = 0$.

[6]See Definition 2.4.

[7]I.e., just "initialize" the clock.

Otherwise, if $\{u_i : i \in n\}$, where $n \subseteq \omega$,[8] is the full set of i.s. of u, then $\|u\| = \sup^+\{u_i : i \in n\}$.[9] Thus, if $n \in \omega$, then $\|u\| = 1 + \max\{\|u_0\|, \ldots, \|u_{n-1}\|\}$.

Note. The *semantics* that makes the definition of depths meaningful is that type-1 oracles are deterministic. Thus, the time it takes to compute $\alpha(y)$ is fully determined by α and y. We may wish to extend the above definition to include tuples u that are well-formed to be "computations" (i.e., in $\Pi_{\mathbb{I}}$) but fail to be so because they are "divergent". For such u we may set $\|u\| = \aleph_1$.[10] As usual, one defines

Definition 2.5. A relation R of rank (k, l) is *recursive in* $\mathbb{I}$ iff its characteristic function, given by $\chi(\vec{x}, \vec{\alpha}) =$ **if** $R(\vec{x}, \vec{\alpha})$ **then** 0 **else** 1, is in $\mathcal{R}^{\Pi}_{\mathbb{I}}$. It is *semi-recursive in* $\mathbb{I}$ iff $R(\vec{x}, \vec{\alpha}) = \mathrm{dom}(\{e\}_{\mathbb{I}})$ for some $e \in \omega$.

The following are immediately obtained in the standard manner:

Theorem 2.1. (Kleene's 2nd Recursion Theorem) *If F of rank $(k+1, l)$ is in $\mathcal{P}^{\Pi}_{\mathbb{I}}$, then there is an $e \in \omega$ such that*

$$\{e\}_{\mathbb{I}}(\vec{x}, \vec{\alpha}) = F(e, \vec{x}, \vec{\alpha}) \quad \text{for all} \quad \vec{x}, \vec{\alpha}.$$[11]

Corollary 2.1. *$\mathcal{P}^{\Pi}_{\mathbb{I}}$ is closed under unbounded search, (μy).*

Corollary 2.2. *$\mathcal{P}^{\Pi}_{\mathbb{I}}$ is closed under primitive recursion.*

Corollary 2.3. *The relations semi-recursive in $\mathbb{I}$ are closed under $\wedge$, $(\forall y)$, and $(\forall y)_{\leq z}$.*

Note. Closure under $(\forall y)$ is due to the equivalence "$(\forall y)\{e\}(y, \vec{x}, \vec{\alpha}) \downarrow$ iff $\mathbb{I}\big(\lambda y.\{e\}(y, \vec{x}, \vec{\alpha})\big) \downarrow$". None of the other results in the corollaries above need the presence of $\mathbb{I}$.

Definition 2.6. A partial functional F of rank $(0, 1)$ is *weakly partial recursive in* $\mathbb{I}$ iff there is an (ordinary) primitive recursive function f of rank $(3, 0)$ such that for all $e \in \omega$ and all $\vec{x}$ ($k = \mathrm{length}(\vec{x})$) and $\vec{\alpha}$ ($l = \mathrm{length}(\vec{\alpha})$), $\{f(k, l, e)\}(\vec{x}, \vec{\alpha}) = F(\lambda y.\{e\}_{\mathbb{I}}(y, \vec{x}, \vec{\alpha}))$.

A partial functional of rank $(0, 1)$ contains in its left field all partial functions $\omega \to \omega$. Its domain, of course, could be much smaller. A *total restricted* functional F satisfying Definition 2.6 must also satisfy $F(\lambda y.\{e\}_{\mathbb{I}}(y, \vec{x}, \vec{\alpha})) \downarrow$ iff $\lambda y.\{e\}_{\mathbb{I}}(y, \vec{x}, \vec{\alpha})$ is total. Such a functional will be called just *weakly recursive*, dropping the qualification "partial" (hoping that confusion will not ensue).

In Definition 2.6 one normally asks for an additional condition, on subcomputations of the $\{e\}_{\mathbb{I}}$, but we will not need this here. It is immediate from Definition 2.2 that $\mathbb{I}$ is weakly recursive (in $\mathbb{I}$) since $\mathbb{I}(\lambda y.\{e\}_{\mathbb{I}}(y, \vec{x}, \vec{\alpha})) = \{\langle 11, k, l, e\rangle\}(\vec{x}, \vec{\alpha})$, for all "parameters" $\vec{x}, \vec{\alpha}$, and $\lambda kle.\langle 11, k, l, e\rangle$ is primitive recursive.

[8] In this notation we think of n as an ordinal less than or equal to ω, i.e., if $0 \neq n \neq \omega$, then $n = \{0, 1, \ldots, n-1\}$.

[9] For any set of ordinals $\{\kappa : \ldots\}$, we let $\sup^+\{\kappa : \ldots\}$ mean the *least upper bound* of $\{\kappa + 1 : \ldots\}$, following standard set-theoretic notation.

[10] The rule-set used in the recursive definition of $\Pi_{\mathbb{I}}$ is $\aleph_1$-based, i.e., all formation rules have premises with strictly fewer than $\aleph_1$ elements. Thus, every computation $u \in \Pi_{\mathbb{I}}$ satisfies $\|u\| < \aleph_1$ ("all depths are finite or enumerable ordinals"). Hence, $\aleph_1$ is appropriate notation to denote "infinity"—in other words non-membership in $\Pi_{\mathbb{I}}$ (or divergence of computation).

[11] Throughout this paper "=" denotes Kleene's "weak equality", that is, $f(\sigma) = g(\tau)$ iff $f(\sigma) \uparrow \wedge g(\tau) \uparrow \vee (\exists x)(f(\sigma) = x \wedge g(\tau) = x)$.

Definition 2.7. (Quantification over ω) We define a *total restricted* functional E_ω by

$$E_\omega(\alpha) = \begin{cases} 0 & \textbf{if } (\forall n \in \omega)\alpha(n) \downarrow \wedge (\exists n \in \omega)\alpha(n) = 0 \\ 1 & \textbf{if } (\forall n \in \omega)(\exists m \in \omega)(\alpha(n) = m \wedge m > 0) \end{cases}$$

Theorem 2.2. (p-normality) *Assume that E_ω is weakly recursive in $\mathbb{I}$. Then, there is a functional H in $P^{\Pi}_{\mathbb{I}}$ satisfying*

(a) $\|x\| < \aleph_1 \vee \|y\| < \aleph_1$ implies $H(x, y, \vec{\alpha}) \downarrow$,

(b) $\|x\| < \aleph_1 \wedge \|x\| \leq \|y\|$ implies $H(x, y, \vec{\alpha}) = 0$,

(c) $\|x\| > \|y\|$ implies $H(x, y, \vec{\alpha}) = 1$.

Here the type-1 part of both truncated *computations $x = \langle s, \sigma\rangle$ and $y = \langle t, \tau\rangle$ is $\vec{\alpha}$ with $l = length(\vec{\alpha})$, (σ, τ are the respective type-0 input sequences).*

Proof:

The proof is standard. See for example [2] for a detailed account in the context where type-1 inputs are total, or [1, 4] for a proof-sketch that involves only the "interesting cases" (these latter two works also only deal with total type-1 inputs).

We too only confine ourselves to a few interesting cases, one of which involves computations that evaluate a (partial) type-1 input ($\alpha(x_i)$). The latter are troublesome in the standard Kleene-schemata setting, *if* α is allowed to be non-total, for they make H non-monotone (see introductory remarks in [6]), causing the proof to break down. Here, in the presence of the "clock axiom", non-monotonicity is not a problem.[12]

We define $\lambda xy\vec{\alpha}.H(x, y, \vec{\alpha})$ by cases. The recursive definition of H is based on the observation (see Definition 2.4):

$$\begin{aligned} &\text{if } \|x\| < \aleph_1, \text{ then } \|x\| \leq \|y\| \text{ iff} \\ &\quad (\forall x')\Big(x' \text{ is i.s. of } x \rightarrow (\exists y')(y' \text{ is i.s. of } y \wedge \|x'\| \leq \|y'\|)\Big) \end{aligned}$$

and therefore

$$\begin{aligned} &\|x\| > \|y\| \text{ iff} \\ &(\exists x')\Big(x' \text{ is i.s. of } x \wedge (\forall y')(y' \text{ is i.s. of } y \rightarrow \|x'\| > \|y'\|)\Big) \end{aligned}$$

Here are some interesting cases (we omit all the tedious but straightforward formalities):
(i) Let $x = \langle\langle 7, k + m + 1, l, m\rangle, a, \vec{b}_m, \vec{x}_k\rangle$ and $y = \langle\langle 7, k' + m' + 1, l, m'\rangle, c, \vec{d}_{m'}, \vec{y}_{k'}\rangle$ where we have omitted the $\vec{\alpha}$-part for typographical convenience. The i.s. of x are $\langle b_i, \vec{x}_k\rangle$, $i = 1, \ldots, m$ and $\langle a, \{b_1\}(\vec{x}_k), \ldots, \{b_m\}(\vec{x}_k)\rangle$ (of course, one, or all of the $\{b_i\}(\vec{x}_k)$ might be undefined). Correspondingly, the i.s. of y are $\langle d_i, \vec{y}_{k'}\rangle$, $i = 1, \ldots, m'$ and $\langle c, \{d_1\}(\vec{y}_{k'}), \ldots, \{d_{m'}\}(\vec{y}_{k'})\rangle$. For the sake of notational convenience let us name all the above i.s., in the order they were written, by the symbols u_i (for $i = 1, \ldots, m$), u_0, v_i (for $i = 1, \ldots, m'$), v_0. Then, $H(x, y, \vec{\alpha})$ is computed by the following flowchart.

[12] It means however, as in [2], that we must use the 2nd recursion theorem (2.1) in our proof rather than the 1st recursion theorem (used in [1, 4]).

$$
\begin{tikzcd}
\boxed{H(u_0, v_0, \vec{\alpha}) = 0} \arrow[r, "\text{yes}"] \arrow[d, "\text{no}"'] & \cdots \arrow[r, "\text{yes}"] & \boxed{H(u_m, v_0, \vec{\alpha}) = 0} \arrow[r, "\text{yes}"] \arrow[d, "\text{no}"'] & 0 \\
\boxed{H(u_0, v_1, \vec{\alpha}) = 0} \arrow[r, "\text{yes}"] \arrow[d, "\text{no}"'] & \cdots \arrow[r, "\text{yes}"] & \boxed{H(u_m, v_1, \vec{\alpha}) = 0} \arrow[r, "\text{yes}"] \arrow[d, "\text{no}"'] & 0 \\
\vdots \arrow[d, "\text{no}"'] & & \vdots \arrow[d, "\text{no}"'] & \\
\boxed{H(u_0, v_{m'}, \vec{\alpha}) = 0} \arrow[r, "\text{yes}"] \arrow[d, "\text{no}"'] & \cdots \arrow[r, "\text{yes}"] & \boxed{H(u_m, v_{m'}, \vec{\alpha}) = 0} \arrow[r, "\text{yes}"] \arrow[d, "\text{no}"'] & 0 \\
1 & & 1 &
\end{tikzcd}
$$

(ii) Let $x = \langle\langle 7, k+m+1, l, m\rangle, a, \vec{b}_m, \vec{x}_k\rangle$ and $y = \langle\langle 11, k', l, c\rangle, \vec{y}_{k'}\rangle$. We denote the i.s. of x as in (i) above. Let $v_t = \langle c, t, \vec{y}_{k'}\rangle$, for all $t \in \omega$, be the i.s. of y. $H(x, y, \vec{\alpha})$ is given by the flowchart below.

$$
\begin{tikzcd}
\boxed{(\exists t)H(u_0, v_t, \vec{\alpha}) = 0} \arrow[r, "\text{yes}"] \arrow[d, "\text{no}"'] & \cdots \arrow[r, "\text{yes}"] & \boxed{(\exists t)H(u_m, v_t, \vec{\alpha}) = 0} \arrow[r, "\text{yes}"] \arrow[d, "\text{no}"'] & 0 \\
1 & & 1 &
\end{tikzcd}
$$

where $(\exists t)H(u_i, v_t, \vec{\alpha}) = 0$, for $i = 0, \ldots, m$, is implemented as $E_\omega\big(\lambda t.H(u_i, v_t, \vec{\alpha})\big) = 0$.

(iii) Let $x = \langle\langle 6, k, l, i, j\rangle, \vec{x}_k\rangle$ and $y = \langle\langle 6, k', l, i', j'\rangle, \vec{y}_{k'}\rangle$.

Here we have just two i.s., $u = \langle\langle 10, k+1, l, i, j\rangle, 0, \vec{x}_k\rangle$ and $v = \langle\langle 10, k'+1, l, i', j'\rangle, 0, \vec{y}_{k'}\rangle$ respectively. Set $H(x, y, \vec{\alpha}) = H(u, v, \vec{\alpha})$.

(iv) Let $x = \langle\langle 10, k+1, l, i, j\rangle, t, \vec{x}_k\rangle$ and $y = \langle\langle 10, k'+1, l, i', j'\rangle, r, \vec{y}_{k'}\rangle$.

Set $u = \langle\langle 10, k+1, l, i, j\rangle, t+1, \vec{x}_k\rangle$ and $v = \langle\langle 10, k'+1, l, i', j'\rangle, r+1, \vec{y}_{k'}\rangle$. These are the *potential* i.s. of x, y respectively. Then,

$$
H(x, y, \vec{\alpha}) = \begin{cases} H(u, v, \vec{\alpha}) & \textbf{if } \mathbf{X}(t, x_i, \alpha_j) \cdot \mathbf{X}(r, y_{k'}, \alpha_{j'}) = 1 \\ 0 & \textbf{if } \mathbf{X}(t, x_i, \alpha_j) = 0 \\ 1 & \textbf{otherwise} \end{cases}
$$

At the end of all this we have a recursive definition "$H(x, y, \vec{\alpha}) = \cdots H(u, v, \vec{\alpha}) \cdots$". By 2.1, there is an $e \in \omega$ such that $\{e\}(x, y, \vec{\alpha}) = \cdots \{e\}(u, v, \vec{\alpha}) \cdots$, for all $x, y, \vec{\alpha}$, and therefore $H = \{e\}$.

The proof that the inductive definition of H gives us what we want proceeds by a straightforward induction on the ordinal $\min(\|x\|, \|y\|)$, simultaneously for (b)–(c) of the theorem, while (a) follows directly from (b) and (c). (See, for example, [1, 2, 4].) □

Now one gets the Selection Theorem via the standard proof (see any of [1, 2, 4]).

Corollary 2.4. *If E_ω is weakly recursive in $\mathbb{I}$, then there is for each k, l a $\Pi_\mathbb{I}$-computable partial functional $Sel^{(k,l)}$ of rank $(k+1, l)$, such that*
(1) $(\exists y)\{a\}(y, \vec{x}, \vec{\alpha}) \downarrow \leftrightarrow Sel^{(k,l)}(a, \vec{x}, \vec{\alpha}) \downarrow$, *and*
(2) $(\exists y)\{a\}(y, \vec{x}, \vec{\alpha}) \downarrow \to \{a\}(Sel^{(k,l)}(a, \vec{x}, \vec{\alpha}), \vec{x}, \vec{\alpha}) \downarrow$.

From the above, standard techniques yield that the $\Pi_\mathbb{I}$-semi-recursive relations are closed under $\vee, (\exists y)$ and $(\exists y)_{\le z}$ and that a functional is in $P^\Pi_\mathbb{I}$ iff its *graph* is. The latter yields in the obvious way closure of $P^\Pi_\mathbb{I}$ under definition by *positive semi-recursive cases.* Namely, if each f_i is in $P^\Pi_\mathbb{I}$ and each S_i is semi-recursive in $\mathbb{I}$, then if f given by the following equivalence is a function, it is in $P^\Pi_\mathbb{I}$: $y = f(\vec{x}, \vec{\alpha}) \equiv y = f_1(\vec{x}, \vec{\alpha}) \wedge S_1(\vec{x}, \vec{\alpha}) \vee \cdots \vee y = f_k(\vec{x}, \vec{\alpha}) \wedge S_k(\vec{x}, \vec{\alpha})$. It now follows that $R(\vec{x}, \vec{\alpha})$ is recursive in $\mathbb{I}$ iff both $R(\vec{x}, \vec{\alpha})$ and $\neg R(\vec{x}, \vec{\alpha})$ are semi-recursive in $\mathbb{I}$ (for the *if*, define the characteristic function of R by the two positive semi-recursive cases R and $\neg R$).
Note. It is clear that $\Pi \subseteq \Pi_\mathbb{I}$, or $(e, \vec{x}, \vec{\alpha}, y) \in \Pi \to (e, \vec{x}, \vec{\alpha}, y) \in \Pi_\mathbb{I}$. In other words, for all $e \in \omega$, $\{e\} \subseteq \{e\}_\mathbb{I}$. Thus, if $\{e\}$ is total, then $\{e\} = \{e\}_\mathbb{I}$. This yields $\mathcal{R}^\Pi \subseteq \mathcal{R}^\Pi_\mathbb{I}$. We can get a bit more, indeed, we have

Corollary 2.5. *If E_ω is weakly recursive in $\mathbb{I}$, then $\mathcal{P}^\Pi \subseteq \mathcal{P}^\Pi_\mathbb{I}$.*

Proof:
Let $f \in \mathcal{P}^\Pi$. Then $\lambda y\vec{x}\vec{\alpha}.y = f(\vec{x}, \vec{\alpha})$ is semi-recursive in the unrelativized sense.[13] By the "weak" normal form theorem of [6][14] in the unrelativized theory, there is a recursive L such that, for some e, $y = f(\vec{x}, \vec{\alpha}) \equiv (\exists z)L(\langle e, y, \vec{x}\rangle, z, \vec{\alpha}) = 0$.

By the preceding note, the predicate quantified by $(\exists z)$ is in $\mathcal{R}^\Pi_\mathbb{I}$, thus the left hand side of $\equiv$ is semi-recursive in $\mathbb{I}$. Therefore, $f \in \mathcal{P}^\Pi_\mathbb{I}$. □

3. Acknowledgements

I wish to thank the referee whose suggestions improved the clarity of the paper, in particular in connection with the introduction of $\mathbf{X}$ in Section 2. The referee also pointed out that the presence of the "clock", $\mathbf{X}$, makes it possible to simulate computations relative to a partial function α by computations relative to some total function β. One introduces the latter by first letting

$$\theta(t, x) = \textbf{if } \mathbf{X}(t, x, \alpha) = 0 \textbf{ then } \alpha(x) + 1 \textbf{ else } 0$$

from which α and $\mathbf{X}$ can be recovered (i.e., computed) as

$$\alpha(x) = \theta((\mu t)[\theta(t, x) > 0], x) - 1 \tag{1}$$

and

$$\mathbf{X}(t, x, \alpha) = \textbf{if } \theta(t, x) = 0 \textbf{ then } 1 \textbf{ else } 0 \tag{2}$$

and finally setting $\beta = \lambda x.\theta((x)_0, (x)_1)$.

[13] "Unrelativized" or "absolute" means that the clause for $\mathbb{I}$ is removed from Definition 2.1.
[14] The referee has produced a counterexample to the "*strong*" normal form theorem of [7].

References

[1] Fenstad, J.E.: *General Recursion Theory; An Axiomatic Approach*, New York: Springer-Verlag, 1980.

[2] Hinman, P.G.: *Recursion-Theoretic Hierarchies*, New York: Springer-Verlag, 1978.

[3] Kleene, S.C.: "Recursive functionals and quantifiers of finite type", *Trans. Amer. Math. Soc.*, **91**, 1959, 1–52, **108**, 1963, 106–142.

[4] Moldestad, Johan: *Computations in Higher Types*, New York: Springer-Verlag (Lecture Notes in Mathematics series), 1977.

[5] Moschovakis, Y.N.: "Abstract first order computability", *Trans. Amer. Math. Soc.*, **138**, 1969, 427–464; 465–504.

[6] Tourlakis, G.: "Some reflections on the foundations of ordinary recursion theory and a new proposal", *Zeitschrift f. math. Logik u. Grund. d. Math.*, **32**, 1986, 503–515.

[7] Tourlakis, G.: "Recursion in partial type-1 objects with well-behaved oracles", *Mathematical Logic Quarterly*, **42**, 1996, 449–460.