Recursion on Abstract Structures

by Peter G. Hinman

ABSTRACT. We develop and compare two models for computability over an abstract structure. The first, characterized in term of generalized register machines, provides a good theory for a large class of first-order structures. The second, defined in terms of minimal solutions for functional equations, is more versatile and handles many common second-order examples.

1. Introduction

From its origins as a theory of computable functions of natural numbers, Recursion Theory has been extended and generalized to include many other sorts of 'computable' functions. The first such extensions were directed towards establishing theories of computability over particular domains, such as initial segments of the ordinal numbers or the finite-type structure over the natural numbers. Later work developed theories of computabilty over abstract structures, initially first-order structures, then special sorts of second-order structures. Along the way appeared several still more abstract axiomatic computation theories. Our goal here is to survey some of these developments with an emphasis on the role of inductive definability as the basic principle of general computability. We make no claim to completeness.

One of the striking features of ordinary (or 'classical') Recursion Theory is the diversity of equivalent formal characterizations. Most of these fall into one of three general classes by characterizing a (partial) recursive function(al) as one which

(1) is computable by an abstract machine, such as a Turing or register machine;

(2) is definable in existential form in an interpreted formal language such as first-order arithmetic or an equational calculus;

(3a) belongs to the smallest class of functions which contains certain very simple functions and is closed under a small number of generating principles;

(3b) is represented by objects belonging to some other inductively defined set.

Attempts to develop generalized notions of computability have made use of extended versions of each of these. A full review of these efforts would merit a (long!) article to itself, and we mention here only some of the highlights.

Characterizations of computability based on abstract machines have a natural appeal, although they have not been uniformly successful. For example, a version of Turing machines for recursion in finite types was worked out carefully in [Kl3,4], but this approach was largely neglected by subsequent research in favor of the inductive definability (3b) approach of [Kl1,2]. On the other hand, Friedman's notion [Fr] of a finite algorithmic procedure (FAP) over any first-order structure was elaborated in the articles [MS-HT1 and 2] and [Tu] (or see the excellent summary in [Fe; Chapter 0]) and more recently the article [BSS] characterizes a natural notion of computability over the real numbers — and more generally over any ring — in terms of a kind of register machine. We shall develop this approach in several sections below.

In the book [TuZu1] and the series of papers [TuZu2-5] Tucker and Zucker explore several other notions of computability over abstract algebras. Among other topics these works consider versions of μ-recursion, while-programs, semi-computability, flow charts, infinitary quantifier-free definability, and computable functionals. Some of these build on earlier work of Engeler [En1,2] and Herman and Isard [HeIs]. Most of these notions coincide extensionally with one of those described below.

Definability has been a very successful framework for certain parts of Generalized Recursion Theory. It was early recognized that for an admissible ordinal α, existential definability over the segment L_α of the hierarchy of constructible sets is a natural and useful characterization of the partial α-recursive functions. The point is that just as the computation process for obtaining the value of an ordinary recursive function can be coded as a natural number, so the corresponding process over an admissible ordinal α can be coded as a member of L_α. This idea was extended in the Companion Theorem [Mo1; 9E1] which shows that for any Spector class Γ of relations (an analogue of the semi-recursive relations) over a transitive infinite set A (in particular an infinite ordinal), there exists an admissible set $\mathcal{M} \supseteq A$ such that (among other properties) the members of Γ are exactly the relations Σ_1 definable over $\mathcal{M}$ relative to a single fixed relation. This more complex method is necessary because most structures are not rich enough to code the relevant computations. This approach will not play a big role in this article.

It has turned out that approach (3) has the widest range of application. Not only does it work in more contexts, but it seems more intrinsic in avoiding reference to other constructs such as machines or formulas. The ordinary partial recursive functions over ω form the smallest class which contains some simple functions (for example, the primitive recursive functions) and is closed under composition and search. In developing the theory of recursion for finite types, Kleene (in [Kl1,2]) relied on method (3b) in defining inductively a class $\mathcal{C}$ of tuples of the form (a, x, n), where $a, n \in \omega$ and x is a finite sequence of finite-type objects, and calling a function F partial recursive iff for some (index) a, $F = \{a\}$, where for all x and n,

$$\{a\}(x) \simeq n \quad \Longleftrightarrow \quad (a, x, n) \in \mathcal{C}.$$

The closure conditions for $\mathcal{C}$ correspond to simple initial functions, composition and substitution, and the key principle of data-as-program which introduces the function

$$H(a, x) \simeq \{a\}(x).$$

This approach is convenient for many purposes because it introduces indexing (Gödel numbering) at the beginning of the development and makes available the Second Recursion Theorem as a basic tool for complex computations. The details for finite types and ordinals are worked out in [Hi]. The drawback to this approach is that it relies on the richness of the underlying structure to code the 'programs' as indices belonging to the universe of computation. At the very least this requires a copy of the natural numbers and some kind of coding for finite sequences.

It was the insight of Platek [Pl] that many of these problems are avoided if we take as our basic generating principle the *First* Recursion Theorem instead of the Second. This idea was further refined by Moldestad [Mol], much improved by Kechris and Moschovakis in [Ke-Mo], and further developed in the papers [Mo2,3]. The key property of this approach is that it is inherently second-order; computable functions are generated as least fixed points of computable functionals. This means that computation relative to fixed functions and even functionals is included in the theory from the beginning. This approach will be developed in some detail below.

The plan for the rest of this article is as follows. In Section 2 we set out our basic framework and discuss the simplest functions over a structure, the explicit ones. In Sections 3 and 4 we introduce the two main models of computation based on register machines and recursion. Section 5 contains some examples. In Section 6 we show that much of basic recursion theory generalizes to computation over a structure which can be considered an extension of ordinary arithmetic. Section 7 explores some of the basic properties of semi-recursive relations in an abstract setting, and Section 8 gives two technical proofs.

2. Structures and Functionals

Much of our notation will be relatively but not entirely standard, and we review it here. A ***first-order structure*** is a quadruple $\mathfrak{A} = (A, \mathsf{Rel}_{\mathfrak{A}}, \mathsf{Fn}_{\mathfrak{A}}, \mathsf{Dis}_{\mathfrak{A}})$, where A is a non-empty set, $\mathsf{Rel}_{\mathfrak{A}}$ is a finite (possibly empty) set of finitary ***relations*** on A — that is, subsets of the Cartesian power A^k, for some natural number $k \in \omega$ — $\mathsf{Fn}_{\mathfrak{A}}$ is a finite (possibly empty) set of finitary ***functions*** on A with values in A, and $\mathsf{Dis}_{\mathfrak{A}}$ is a finite subset of A, the set of ***distinguished elements*** of $\mathfrak{A}$. The equality relation $=_A$ will be included explicitly rather than implicitly as is common in first-order logic. Associated with such a structure is a standard ***first-order language*** with non-logical relation, function and constant symbols corresponding to each of relations, functions, and distinguished elements, respectively. The ***terms*** and ***formulas*** of this language are defined as usual. We shall always assume that A has two distinct elements denoted as $\mathsf{0}_{\mathfrak{A}}$ and $\mathsf{1}_{\mathfrak{A}}$, which are either distinguished elements or values of closed terms. Particular structures will be described in the form $(A, R, \ldots, F, \ldots, a \ldots)$.

Let Pf_A^l denote the set of all l-ary partial functions on A — that is, partial maps $f : A^l \to A$. Then a ***finitary functional on A*** is a partial mapping

$$F : A^k \times \mathsf{Pf}_A^{l_0} \times \cdots \times \mathsf{Pf}_A^{l_{n-1}} \to A,$$

for some ***type*** $(k; \boldsymbol{l}) = (k; l_0, \ldots, l_{n-1})$. In particular, a k-ary partial function is also a functional of type $(k;)$. To specify the value of such a functional we often write $F(\boldsymbol{a}, \boldsymbol{f}) \simeq e$. Thus $\boldsymbol{a}$ represents $a_0, \ldots, a_{k-1}$ and $\boldsymbol{f}$ represents $f_0, \ldots, f_{n-1}$ (for unspecified k and n, which are either determined by context or are arbitrary). The relation $\simeq$ is read 'is defined and equal to'.

We shall be concerned exclusively with ***monotone*** functionals, which have the property

$$F(\boldsymbol{a}, \boldsymbol{f}) \simeq e \quad \text{and} \quad \bigwedge_{i<n} [f_i \subseteq g_i] \qquad \Longrightarrow \qquad F(\boldsymbol{a}, \boldsymbol{g}) \simeq e.$$

Here, of course, $f \subseteq g$ means inclusion as a set of ordered pairs or equivalently that g is a function which extends f. The restriction to monotonicity is dictated by our interest in ***deterministically computable*** functionals, which we view as given by algorithms equipped with 'black box' subroutines or oracles for providing values of the function arguments as needed. Thus if the oracles for $\boldsymbol{f}$ provide all the requested values in the computation of $F(\boldsymbol{a}, \boldsymbol{f})$, then the oracles for $\boldsymbol{g}$ will provide exactly the same values and hence yield the same computation.

A ***functional structure*** is a pair $\mathfrak{A} = (A, \mathsf{Fnl}_{\mathfrak{A}})$, whare $\mathsf{Fnl}_{\mathfrak{A}}$ is a finite set of monotone finitary functionals on A. We consider every first-order structure $\mathfrak{A} = (A, \mathsf{Rel}_{\mathfrak{A}}, \mathsf{Fn}_{\mathfrak{A}}, \mathsf{Dis}_{\mathfrak{A}})$ to be also a functional structure by identifying it with the functional structure whose functionals are: the characteristic function χ_R, a functional of type $(k;)$, for each $R \in \mathsf{Rel}_{\mathfrak{A}}$; the members of $\mathsf{Fn}_{\mathfrak{A}}$, which are already functionals of some type $(k;)$; and a constant functional of type $(0;)$ for each $a \in \mathsf{Dis}_{\mathfrak{A}}$.

As examples and for future reference, we consider several more or less familiar functionals and functional structures.

(1) Probably the most basic example is the ***application*** or ***evaluation functional*** of type $(k; k)$ which exercises the call to the oracle for f:

$$\mathsf{Ev}^{k,k}(\boldsymbol{a}, f) \simeq f(\boldsymbol{a}).$$

(2) Over the set ω of natural numbers we have the ***search functional*** of type (0;1):

$$\begin{aligned} \mathsf{Se}_\omega(g) &\simeq \text{'least' } n\, [g(n) \simeq 0] \\ &\simeq n \quad \Longleftrightarrow \quad g(n) \simeq 0 \quad \text{and} \quad (\forall p < n)\, g(p) \downarrow \neq 0. \end{aligned}$$

Here $g(p) \downarrow \neq 0$ is read '$g(p)$ is defined and different from 0' and means that for some $q \neq 0$, $g(p) \simeq q$.

(3) If H is a functional of type $(k+l;\cdots)$, then for each $\boldsymbol{a}$ and $\boldsymbol{g}$, $\lambda\,\boldsymbol{b}\,.\,H(\boldsymbol{a},\boldsymbol{b},\boldsymbol{g})$ denotes the function $\boldsymbol{b}\mapsto H(\boldsymbol{a},\boldsymbol{b},\boldsymbol{g})$ of type $(l;)$. Thus if G is of type $(k;l,\cdots)$, then

$$F(\boldsymbol{a},\boldsymbol{g}) \simeq G(\boldsymbol{a},\lambda\,\boldsymbol{b}\,.\,H(\boldsymbol{a},\boldsymbol{b},\boldsymbol{g}),\boldsymbol{g})$$

is of type $(k;\cdots)$. A class $\mathcal{F}$ of functionals has the ***substitution property*** or is ***closed under substitution*** iff $G,H\in\mathcal{F}$ implies also $F\in\mathcal{F}$ (together with more general instances as described below). Substitution is a powerful property of a class $\mathcal{F}$ since it implies other closure properties. For example, if $\mathcal{F}$ is a class of functionals over ω with the substitution property and $\mathsf{Se}_\omega\in\mathcal{F}$, then for any $G\in\mathcal{F}$, the functional

$$F(\boldsymbol{m},\boldsymbol{f}) \simeq \text{`least'}\, n\,[G(\boldsymbol{m},n,\boldsymbol{f})\simeq 0] \simeq \mathsf{Se}_\omega(\lambda\,n\,.\,G(\boldsymbol{m},n,\boldsymbol{f}))$$

is also in $\mathcal{F}$. In other words, $\mathcal{F}$ is closed under search over ω.

(4) Another nice example of a particular functional whose main importance is to join with substitution to guarantee a closure property is the ***primitive recursion*** functional over ω of type (2;1):

$$\mathsf{Prim}_\omega(p,n,h) \simeq \begin{cases} n, & \text{if } p=0;\\ h(\mathsf{Prim}_\omega(p-1,n,h)), & \text{otherwise.}\end{cases}$$

Thus if $\mathsf{Prim}_\omega, G, H, \in\mathcal{F}$, $\mathcal{F}$ is closed under substitution, and

$$F(p,\boldsymbol{m},\boldsymbol{f}) \simeq \begin{cases} G(\boldsymbol{m},\boldsymbol{f}), & \text{if } p=0;\\ H(F(p-1,\boldsymbol{m},\boldsymbol{f}),p,\boldsymbol{m},\boldsymbol{f}). & \text{otherwise,}\end{cases}$$

then $F(p,\boldsymbol{m},\boldsymbol{f})\simeq\mathsf{Prim}_\omega(p,G(\boldsymbol{m},\boldsymbol{f}),\lambda\,q\,.\,H(q,p,\boldsymbol{m},\boldsymbol{f}))$, so also $F\in\mathcal{F}$.

(5) Let κ be a limit ordinal. The ***supremum functional*** over κ of type (1;1) is

$$\begin{aligned}\mathsf{Sup}_\kappa(\rho,f) &\simeq \mathrm{Sup}^+_{\pi<\rho} f(\pi)\\ &\simeq \nu<\kappa \iff (\forall\pi<\rho)\,f(\pi)\downarrow<\nu \quad\text{and}\quad (\forall\mu<\nu)(\exists\pi<\rho)f(\pi)\geq\mu.\end{aligned}$$

Note that even if f is a function on κ defined for all $\pi<\rho$, $\mathsf{Sup}_\kappa(\rho,f)$ is undefined when $\{\,f(\pi):\pi<\rho\,\}$ is cofinal in κ. The characteristic property of recursion over recursively regular or admissible ordinals κ is that if f is partial κ-recursive and defined for all $\pi<\rho$, then $\mathsf{Sup}_\kappa(\rho,f)$ is defined. We shall see below that the usual notion of partial κ-recursion can be characterized as recursion over the functional structure $(\kappa,\,\mathsf{Sc},\,0,\,<,\,\mathsf{Sup}_\kappa,\mathsf{Se}_\kappa)$, where Sc is the usual successor function $\mu\mapsto\mu+1$ and

$$\begin{aligned}\mathsf{Se}_\kappa(g) &\simeq \text{`least'}\,\nu\,[g(\nu)\simeq 0]\\ &\simeq \nu \iff g(\nu)\simeq 0 \quad\text{and}\quad (\forall\pi<\nu)\,g(\pi)\downarrow\neq 0.\end{aligned}$$

(6) Let A be a transitive set of sets. The ***replacement functional*** over A of type (1;1) is

$$\begin{aligned}\mathsf{Repl}_A(a,f) &\simeq \{\,f(b):b\in a\,\}_A\\ &\simeq c\in A \iff (\forall b\in a)f(b)\downarrow\in c \quad\text{and}\quad (\forall d\in c)(\exists b\in a)f(b)\simeq d.\end{aligned}$$

The notion of E-recursion over A will be seen below to be characterized as recursion over the structure $\mathfrak{A} = (A, \emptyset, \subseteq, \mathsf{pair}, \cup, \mathsf{Repl}_A)$ and A is E-closed iff for any $a \in A$ and $\mathfrak{A}$-recursive function f, if $f(b)$ is defined for all $b \in a$, then also $\mathsf{Repl}_A(a, f)$ is defined.

(7) Over any set A there is the functional $\mathsf{E}^\sharp_A$ of type (0;1):

$$\mathsf{E}^\sharp_A(f) \simeq \begin{cases} 1_{\mathfrak{A}}, & \text{if } (\exists a \in A)\, f(a) \simeq 1_{\mathfrak{A}}; \\ 0_{\mathfrak{A}}, & \text{if } (\forall a \in A)\, f(a) \simeq 0_{\mathfrak{A}}. \end{cases}$$

We shall see in Section 5 that if $\mathfrak{A}$ is a first-order structure, then recursion over $(\mathfrak{A}, =_A, \mathsf{E}^\sharp_A)$ corresponds to first-order inductive definability over $\mathfrak{A}$.

The first class of functionals that we associate with a functional structure consists of those that are explicitly defined from the basic ones in $\mathsf{Fnl}_{\mathfrak{A}}$.

2.1 Definition. The class $\mathsf{Expl}_{\mathfrak{A}}$ of $\mathfrak{A}$***-explicit functionals*** is the smallest class of functionals over $\mathfrak{A}$ such that

(i) $\mathsf{Expl}_{\mathfrak{A}}$ contains all of the ***initial functionals***:
  (a) $\mathsf{Pr}^{k,l}_i(\boldsymbol{a}, \boldsymbol{f}) = a_i \quad (i < k)$;
  (b) $\mathsf{Ev}^{k,l}_j(\boldsymbol{a}, \boldsymbol{f}) \simeq f_j(\boldsymbol{a}) \quad (j < n,\, l_j = k)$;
  (c) all members of $\mathsf{Fnl}_{\mathfrak{A}}$.

(ii) $\mathsf{Expl}_{\mathfrak{A}}$ is closed under ***composition***, ***substitution***, and ***definition by cases***:
  (a) for any $G, H_0, \ldots, H_{m-1} \in \mathsf{Expl}_{\mathfrak{A}}$, if

$$F(\boldsymbol{a}, \boldsymbol{f}) \simeq G(H_0(\boldsymbol{a}, \boldsymbol{f}), \ldots, H_{m-1}(\boldsymbol{a}, \boldsymbol{f}), \boldsymbol{f})),$$

  then also $F \in \mathsf{Expl}_{\mathfrak{A}}$;
  (b) for any $G, H_0, \ldots, H_{m-1} \in \mathsf{Expl}_{\mathfrak{A}}$, if

$$F(\boldsymbol{a}, \boldsymbol{f}) \simeq G(\boldsymbol{a}, \lambda \boldsymbol{b}^0 \,.\, H_0(\boldsymbol{a}^0, \boldsymbol{b}^0, \boldsymbol{f}), \ldots, \lambda \boldsymbol{b}^{m-1} \,.\, H_{m-1}(\boldsymbol{a}^{m-1}, \boldsymbol{b}^{m-1}, \boldsymbol{f})),$$

  where $\boldsymbol{a}^0, \ldots, \boldsymbol{a}^{m-1}$ are (possibly empty) lists selected from $\boldsymbol{a}$, then also $F \in \mathsf{Expl}_{\mathfrak{A}}$;
  (c) for any G, H, and $I \in \mathsf{Expl}_{\mathfrak{A}}$, if

$$F(c, \boldsymbol{a}, \boldsymbol{f}) \simeq \begin{cases} G(\boldsymbol{a}, \boldsymbol{f}), & \text{if } c = 1_{\mathfrak{A}}; \\ H(\boldsymbol{a}, \boldsymbol{f}), & \text{if } c = 0_{\mathfrak{A}} \\ I(\boldsymbol{a}, \boldsymbol{f}), & \text{otherwise;} \end{cases}$$

  then also $F \in \mathsf{Expl}_{\mathfrak{A}}$.

We note that easily

2.2 Proposition. $\mathsf{Expl}_{\mathfrak{A}}$ *is closed under* ***argument rearrangement*** *and* ***expansion*** *— that is, for any* $G \in \mathsf{Expl}_{\mathfrak{A}}$*, if*

$$F(a_0, \ldots, a_{k-1}, f_0, \ldots, f_{l-1}) \simeq G(a_{i_0}, \ldots, a_{i_{m-1}}, f_{j_0}, \ldots f_{j_{n-1}}),$$

where $i_0, \ldots, i_{m-1} < k$ *and* $j_0, \ldots, j_{n-1} < l$*, then also* $F \in \mathsf{Expl}_{\mathfrak{A}}$*.*

Proof. We illustrate the proof with an example: if $G \in \mathsf{Expl}_{\mathfrak{A}}$ and $F(a, b, c, f, g) \simeq G(c, b, g)$, with f and g of type $(k;)$ and $(l;)$, respectively, then set

$$\begin{aligned} H(c, b, \boldsymbol{d}, f, g) &\simeq \mathsf{Ev}_1^{k,k,l}(\mathsf{Pr}_2^{k+2,k,l}(c, b, \boldsymbol{d}, f, g), \ldots, \mathsf{Pr}_{k+1}^{k+2,k,l}(c, b, \boldsymbol{d}, f, g), f, g) \\ &\simeq g(\boldsymbol{d}), \end{aligned}$$

$$G_0(c, b, f, g) \simeq G(c, b, g) \simeq G(c, b, \lambda\, \boldsymbol{d}\,.\, H(c, b, \boldsymbol{d}, f, g)).$$

Then $H, G_0 \in \mathsf{Expl}_{\mathfrak{A}}$ and since

$$F(a, b, c, f, g) \simeq G_0(\mathsf{Pr}_2^{3,k,l}(a, b, c, f, g), \mathsf{Pr}_1^{3,k,l}(a, b, c, f, g), f, g),$$

also $F \in \mathsf{Expl}_{\mathfrak{A}}$. □

2.3 Remark. It is easy to derive that the functionals

$$\mathsf{Cmp}^{k,m}(\boldsymbol{a}, g, h_0, \ldots, h_{m-1}) \simeq g(h_0(\boldsymbol{a}), \ldots, h_{m-1}(\boldsymbol{a}))$$

and

$$\mathsf{Cases}^k(c, \boldsymbol{a}, g, h, i) \simeq \begin{cases} g(\boldsymbol{a}), & \text{if } c = 1_{\mathfrak{A}}; \\ h(\boldsymbol{a}), & \text{if } c = 0_{\mathfrak{A}}; \\ i(\boldsymbol{a}), & \text{otherwise;} \end{cases}$$

are $\mathfrak{A}$-explicit. Conversely, any class of functionals which includes these functionals and is closed under substitution (in the form given above) is also closed under composition and definition by cases, since, for example,

$$G(H(\boldsymbol{a}, \boldsymbol{f}), \boldsymbol{f}) \simeq \mathsf{Cmp}^{k,1}(\boldsymbol{a}, \lambda\, b\,.\, G(b, \boldsymbol{f}), \lambda\, \boldsymbol{c}\,.\, H(\boldsymbol{c}, \boldsymbol{f})).$$

Although the placement of variables in the schemas for composition and substitution is precise and somewhat arbitrary, by argument rearrangement and expansion (2.2), all reasonable variations are derivable. For example, if G, H, and I are $\mathfrak{A}$-explicit and

$$F(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{f}, \boldsymbol{g}) \simeq G(\boldsymbol{a}, H(\boldsymbol{a}, \boldsymbol{g}), \lambda\, \boldsymbol{c}\,.\, I(\boldsymbol{b}, \boldsymbol{c}, \boldsymbol{f}), \boldsymbol{g}, \boldsymbol{f}),$$

then also F is $\mathfrak{A}$-explicit. Similarly, we have closure under much more general schemas of definition by cases — for example, if G, H, I, J, and K are $\mathfrak{A}$-explicit (or partial $\mathfrak{A}$-recursive), J and K are total functions, and

$$F(\boldsymbol{a}, \boldsymbol{f}) \simeq \begin{cases} G(\boldsymbol{a}, \boldsymbol{f}), & \text{if } J(\boldsymbol{a}) = 1_{\mathfrak{A}}; \\ H(\boldsymbol{a}, \boldsymbol{f}), & \text{if } J(\boldsymbol{a}) = 0_{\mathfrak{A}} \quad \text{and} \quad K(\boldsymbol{a}) = 1_{\mathfrak{A}}; \\ I(\boldsymbol{a}, \boldsymbol{f}), & \text{otherwise,} \end{cases}$$

then also F is $\mathfrak{A}$-explicit. We shall use these extensions of the basic schemas without mention. In particular, we will need the following functions which are $\mathfrak{A}$-explicit for every $\mathfrak{A}$:

$$\mathsf{Not}(a) \simeq \begin{cases} \mathsf{0}_{\mathfrak{A}}, & \text{if } a = \mathsf{1}_{\mathfrak{A}}; \\ \mathsf{1}_{\mathfrak{A}}, & \text{if } a = \mathsf{0}_{\mathfrak{A}}; \end{cases}$$

$$\mathsf{And}(a,b) \simeq \begin{cases} \mathsf{1}_{\mathfrak{A}}. & \text{if } a = \mathsf{1}_{\mathfrak{A}} = b; \\ \mathsf{0}_{\mathfrak{A}}, & \text{if } a = \mathsf{0}_{\mathfrak{A}} \text{ or } b = \mathsf{0}_{\mathfrak{A}}; \end{cases}$$

$$\mathsf{Or}(a,b) \simeq \begin{cases} \mathsf{1}_{\mathfrak{A}}. & \text{if } a = \mathsf{1}_{\mathfrak{A}} \text{ or } b = \mathsf{1}_{\mathfrak{A}}; \\ \mathsf{0}_{\mathfrak{A}}, & \text{if } a = \mathsf{0}_{\mathfrak{A}} = b. \end{cases}$$

$$\mathsf{IsZero}_{\mathfrak{A}}(a) \simeq \begin{cases} \mathsf{1}_{\mathfrak{A}}, & \text{if } a = \mathsf{0}_{\mathfrak{A}}; \\ \mathsf{0}_{\mathfrak{A}}, & \text{otherwise} \end{cases}$$

$$\mathsf{IsOne}_{\mathfrak{A}}(a) \simeq \begin{cases} \mathsf{1}_{\mathfrak{A}}, & \text{if } a = \mathsf{1}_{\mathfrak{A}}; \\ \mathsf{0}_{\mathfrak{A}}, & \text{otherwise} \end{cases}$$

Note that composing these functions with total functions gives the expected result, but while

$$\mathsf{And}(F(x), G(x)) \downarrow \quad \Longleftrightarrow \quad F(x) \downarrow \quad \text{and} \quad G(x) \downarrow,$$

the analogous equivalence for Or is not in general true. Indeed, we shall see in Section 7 that although the class of $\mathfrak{A}$-semi-recursive relations is always closed under intersection, an additional hypothesis is needed to guarantee closure under union.

2.4 Proposition. *If* $\mathfrak{A}' = (A, \mathsf{Fnl}_{\mathfrak{A}'})$, *and* $\mathsf{Fnl}_{\mathfrak{A}} \subseteq \mathsf{Fnl}_{\mathfrak{A}'} \subseteq \mathsf{Fnl}_{\mathfrak{A}} \cup \mathsf{Expl}_{\mathfrak{A}}$, *then* $\mathsf{Expl}_{\mathfrak{A}} = \mathsf{Expl}_{\mathfrak{A}'}$.

Proof. Immediate. □

We introduce one other notion here for later reference:

2.5 Definition. A functional F is ***continuous*** iff whenever $F(\boldsymbol{a}, \boldsymbol{f}) \simeq e$, there exist finite $f_0' \subseteq f_0, \ldots, f_{n-1}' \subseteq f_{n-1}$ such that also $F(\boldsymbol{a}, \boldsymbol{f}') \simeq e$.

2.6 Proposition. *For any functional structure* $\mathfrak{A}$, *if every* $F \in \mathsf{Fnl}_{\mathfrak{A}}$ *is continuous, then also every* $F \in \mathsf{Expl}_{\mathfrak{A}}$ *is continuous.*

Proof. A straightforward induction. □

3. Register machines over first-order structures

For all of ths section we restrict $\mathfrak{A}$ to be a first-order structure. We recall the notion of an ***unlimited register machine*** (URM) due to Shepherdson and Sturgis [Sh-St]; we follow essentially the excellent presentation of Cutland [Cu]. The machine M has an infinite sequence $R_0, R_1, \ldots$ of ***registers*** (memory cells) each of which contains at each discrete time step a natural number $r_0, r_1, \ldots$. At the beginning of a computation all $r_p = 0$. A ***program*** $\pi = (I_0, \ldots I_m)$ consists of a finite sequence of ***instructions***, each of one of the following four types:

$r_p := r_p + 1$

$r_p := 0$

$r_p := r_q$

if $r_p = r_q$, go to I_s, else go to I_t.

A computation $\pi(m_0, \ldots, m_{k-1})$ proceeds by setting $r_p := m_p$ for $p < k$ and $r_p := 0$ for $p \geq k$ and executing the instructions starting with I_0 in numerical order except as dictated by a conditional instruction. The computation terminates if and when the machine would attempt to execute a non-existent instruction, either as the result of a conditional instruction or by executing the last instruction of the program, which is not a conditional instruction. If it terminates, the computation has output r_0. Thus every program π computes, for each k, a k-ary partial function (also denoted) π defined by: $\pi(m_0, \ldots, m_{k-1}) \simeq n \iff \pi(m_0, \ldots, m_{k-1})$ terminates with $r_0 = n$. The functions so computed are exactly the partial recursive functions.

Note that a computation $\pi(m_0, \ldots, m_{k-1})$ addresses only the finitely many registers mentioned in the program π so can be considered to take place on a machine with only finitely many registers. It is easy to extend this formalism to compute also the partial recursive functionals by introducing new instructions of the form

$r_p := f_j(r_{q_0}, \ldots, r_{q_{l_j - 1}}) \quad (j < n).$

Note that although a program may be considered to act on a sequence of numerical arguments of any length, the type $\boldsymbol{l} = (l_0, \ldots, l_{n-1})$ of its function arguments is determined by the program.

For any first-order structure $\mathfrak{A} = (A, \mathsf{Fn}_{\mathfrak{A}}, \mathsf{Rel}_{\mathfrak{A}}, \mathsf{Dis}_{\mathfrak{A}})$, we define an ***unlimited register machine over*** $\mathfrak{A}$ ($\mathrm{URM}_{\mathfrak{A}}$) of type $\boldsymbol{l}$ as follows. Each register now contains at each time step an element of A. A program $\pi = (I_0, \ldots, I_m)$ consists of a finite sequence of instructions, each of one of the following five types:

$r_p := f(r_{q_0}, \ldots, r_{q_{k-1}})$ for a k-ary $f \in \mathsf{Fn}_{\mathfrak{A}}$

$r_p := a$ for $a \in \mathsf{Dis}_{\mathfrak{A}}$

$r_p := f_j(r_{q_0}, \ldots, r_{q_{l_j - 1}}) \quad (j < n)$

$r_p := r_q$

if $R(r_{q_0}, \ldots, r_{q_{k-1}})$, go to I_s, else go to I_t for a k-ary $R \in \mathsf{Rel}_{\mathfrak{A}}$.

A program π for a $\mathrm{URM}_{\mathfrak{A}}$ of type $\boldsymbol{l}$ computes in the same way for each k a partial functional $\pi : A^k \times \mathsf{Pf}_A^{l_0} \times \cdots \times \mathsf{Pf}_A^{l_{n-1}} \to A$, of type $(k; \boldsymbol{l})$. We call the resulting partial functionals $\mathfrak{A}$ ***-register computable*** and denote the set of them by $\mathsf{Reg}_{\mathfrak{A}}$.

3.1 Corollary. *If $\Omega = (\omega, \mathsf{Sc}, =_\omega, 0)$, then the Ω-register computable functionals are exactly the partial recursive functionals.*

Although the elementary parts of a theory of computability can be established for the register computable functions over any $\mathfrak{A}$, without further assumptions this is a rather weak notion of computability. First we have

3.2 Proposition. *For any first-order structure $\mathfrak{A}$, $\mathsf{Reg}_{\mathfrak{A}}$ is closed under composition, substitution, and definition by cases. Hence all $\mathfrak{A}$-explicit functions are $\mathfrak{A}$-register computable.*

Proof. These are all simple exercises in programming. Closure of the class of register computable functions under definition by cases follows easily using the conditional instruction, and closure under composition can be proved exactly as in [Cu; 2.3.1]. For substitution, consider the special case where $F(\boldsymbol{a}) \simeq G(\boldsymbol{a}, \lambda \boldsymbol{b}\, . H(\boldsymbol{a}, \boldsymbol{b}))$. The program for G contains instructions of the form $r_p := h(r_{q_0}, \ldots, r_{q_{l-1}})$. We construct a program for F by replacing each such instruction by the program for computing H modified to use the input of G together with $r_{q_0}, \ldots, r_{q_{l-1}}$ for its input, otherwise to use registers with indices larger than any used by G, and to store its output value in r_p. $\square$

Over the natural numbers it is easy to show [Cu; 2.5.2] that the register computable functionals are also closed under search and hence include all of the partial recursive functions. The analogous result here will be that the class of register computable functions is closed under least-fixed-point recursion, as defined in the next section. We shall be able to prove this (in Section 6 below) only for structures which contain a copy of the natural numbers. Partly in preparation for this, we sketch here how to assign indices to the register computable functionals over an arbitrary first-order structure.

First, it is a standard exercise to assign Gödel numbers (indices) to programs. Fix a primitive recursive coding $\langle\, m_0, \ldots, m_{k-1} \,\rangle$ of finite sequences of natural numbers with the usual properties. Then assign numbers to $\mathrm{URM}_{\mathfrak{A}}$ instructions as follows:

(i) To "$r_p := f(r_{q_0}, \ldots, r_{q_{k-1}})$" assign the number $\langle\, 0, i, p, q_0, \ldots, q_{k-1} \,\rangle$, where f is the i-th member of $\mathsf{Fn}_{\mathfrak{A}}$;

(ii) to "$r_p := a$" assign $\langle\, 1, i, p \,\rangle$, where a is the i-th member of $\mathsf{Dis}_{\mathfrak{A}}$;

(iii) to "$r_p := f_j(r_{q_0}, \ldots, r_{q_{l_j-1}})$" assign $\langle\, 2, j, p, q_0, \ldots, q_{l_j-1} \,\rangle$;

(iv) to "$r_p := r_q$" assign $\langle\, 3, p, q \,\rangle$;

(v) to "if $R(r_{q_0}, \ldots, r_{q_{k-1}})$, go to I_s, else go to I_t" assign $\langle\, 4, i, q_0, \ldots, q_{k-1}, s, t \,\rangle$, where R is the i-th member of $\mathsf{Rel}_{\mathfrak{A}}$.

Then each program π is assigned a number $\#(\pi)$ which is the code for the finite sequence of the numbers of its instructions. For any natural number b it is primitively recursively computable whether or not b is the number of a program for a $\mathrm{URM}_{\mathfrak{A}}$, and if so the minimum possible type $\boldsymbol{l}$. For each k, let $\{b\}^{\mathfrak{A}}_{\mathsf{Reg}}$ denote the empty function if b is not the number of a $\mathrm{URM}_{\mathfrak{A}}$ program, otherwise the functional of type $(k;\boldsymbol{l})$ computed by the program. Clearly $\left\langle \{b\}^{\mathfrak{A}}_{\mathsf{Reg}} : b \text{ is the number of a program} \right\rangle$ is an enumeration of $\mathsf{Reg}_{\mathfrak{A}}$. The effectiveness of the proof of the preceding proposition yields immediately

3.3 Proposition. *The closure of* $\mathsf{Reg}_{\mathfrak{A}}$ *under composition, substitution, and definition by cases is effective in terms of these indices — that is, for each closure condition there is a primitive recursive function which computes an index for the resulting functional from indices of the component functionals.* □

4. Recursion

In this section we develop the basic properties of the class $\mathsf{Rec}_{\mathfrak{A}}$ of functionals recursive over an arbitrary functional structure $\mathfrak{A}$. The fundamental generating principle is a version of the First Recursion Theorem, which asserts the existence of computable fixed points for computable functionals.

Consider first a monotone functional I of type $(k;k)$ over a set A. We call a k-ary partial function i a ***fixed point*** of I iff for all $\boldsymbol{a} \in A$,

$$I(\boldsymbol{a}, i) \simeq i(\boldsymbol{a}).$$

The terminology derives from regarding I as a mapping on functions: $I[i] = \lambda\, \boldsymbol{a}\, .\, I(\boldsymbol{a}, i)$. The monotonicity of I guarantees that I has, in fact, a ***least fixed point***

$$I^{\infty} = \bigcap \{\, i : I[i] \subseteq i \,\}.$$

Clearly if I^{∞} is a fixed point at all, then it is least (under inclusion). To see that I^{∞} *is* a fixed point, note that for any i such that $I[i] \subseteq i$, we have by definition $I^{\infty} \subseteq i$, so by monotonicity, $I[I^{\infty}] \subseteq I[i] \subseteq i$. Hence, since I^{∞} is the intersection of all such i, also $I[I^{\infty}] \subseteq I^{\infty}$. Then, again by monotonicity, $I[I[I^{\infty}]] \subseteq I[I^{\infty}]$, so that $I[I^{\infty}]$ is itself such an i and hence $I^{\infty} \subseteq I[I^{\infty}]$. Thus $I^{\infty} = I[I^{\infty}]$.

The function I^{∞} may also be described 'from below' by ***iterating*** I starting with the empty function. For each ordinal σ, set

$$I^{<\sigma} = \bigcup_{\tau<\sigma} I^{\tau} \qquad \text{and} \qquad I^{\sigma} = I[I^{<\sigma}].$$

Thus $I^{<0} = \emptyset$ and an easy induction shows that for all σ, $I^{<\sigma} \subseteq I^{\sigma}$ and that if $I^{<\sigma} = I^{\sigma}$, then also $I^{\sigma} = I^{\rho}$ for all $\rho > \sigma$. Since the domain of each I^{σ} is a subset of A^k, so of power $\mathsf{Card}(A)$, it follows easily that $I^{<\bar{\sigma}} = I^{\bar{\sigma}}$ for some $\bar{\sigma} < \mathsf{Card}(A)^+$, whence $I^{\bar{\sigma}}$ is a fixed point of I. Another straightforward induction shows that $I^{\sigma} \subseteq I^{\infty}$ for all σ, so since I^{∞} is the *least* fixed point of I, necessarily $I^{\bar{\sigma}} = I^{\infty}$. The least such $\bar{\sigma}$ is called the ***closure ordinal*** of I. As an important special case we have

4.1 Proposition. *If I is continuous, then the closure ordinal of I is finite or ω.*

Proof. Continuity easily implies that $I[I^{<\omega}] = I^{<\omega}$. □

The general notion of fixed point goes beyond this only in the inclusion of parameters. We call a functional I ***iterable*** iff it is of some type $(k+l; l_0, \ldots, l_{n-1}, k)$ and a partial function i is a ***fixed point of*** I ***with respect to*** (parameters) $x = (\boldsymbol{b}, \boldsymbol{f})$ iff

$$I(\boldsymbol{a}, x, i) \simeq i(\boldsymbol{a}).$$

For any functional H and sequences of parameters x and y we define H_x by $H_x(y) := H(x, y)$ and call F a ***fixed point of*** I iff for each x, F_x is a fixed point of I with respect to x (or equivalently a fixed point of I_x) — in symbols,

$$I(\boldsymbol{a}, x, F_x) \simeq I_x(\boldsymbol{a}, F_x) \simeq F_x(\boldsymbol{a}) \simeq F(\boldsymbol{a}, x).$$

The ***least fixed point*** I^∞ of I is simply the unique functional such that for each x, I^∞_x is the least fixed point of I_x. The corresponding iterative description of I^∞ is then

$$I^{<\sigma} = \bigcup_{\tau<\sigma} I^\tau \qquad \text{and} \qquad I^\sigma(\boldsymbol{a}, x) \simeq I(\boldsymbol{a}, x, I^{<\sigma}_x).$$

As above, for each x there exists an ordinal $\sigma_x < \mathsf{Card}(A)^+$ such that $I^{<\sigma_x}_x = I^{\sigma_x}_x = I^\infty_x$, and hence an ordinal $\bar{\sigma} \leq \mathsf{Card}(A)^+$ such that $I^{<\bar{\sigma}} = I^{\bar{\sigma}} = I^\infty$.

Finally, we introduce the notion of simultaneous fixed points. Call a sequence $(I_0, \ldots, I_n)$ ***iterable*** iff each I_p is of a type of the form $(k_p + l; l_0, \ldots, l_{m-1}, k_0, \ldots, k_n)$. A sequence of functionals $(F_0, \ldots, F_n)$ is a ***simultaneous fixed point*** of $(I_0, \ldots, I_n)$ iff for all x and $p = 0, \ldots, n$

$$I_p(a_0, \ldots, a_{k_p-1}, x, F_{0,x}, \ldots, F_{n,x}) \simeq F_p(a_0, \ldots, a_{k_p-1}, x).$$

An easy extension of the arguments above shows that there is a unique sequence $(I^\infty_0, \ldots, I^\infty_n)$ which is a ***simultaneous least fixed point*** of $(I_0, \ldots, I_n)$ in the sense that for any other simultaneous fixed point $(F_0, \ldots, F_n)$, we have $I^\infty_p \subseteq F_p$ for all $p \leq n$. The functionals I^∞_p are approximated from below by functionals I^σ_p which satisfy the equations

$$I^\sigma_p(a_0, \ldots, a_{k_p-1}, x) \simeq I_p(a_0, \ldots, a_{k_p-1}, x, I^{<\sigma}_{0,x}, \ldots, I^{<\sigma}_{n,x}).$$

We call I^∞_0 the ***principal least fixed point*** of $I_0, \ldots, I_n$ and write

$$I^\infty_0 = \mathsf{Rec}(I_0, \ldots, I_n).$$

4.2 Definition. The class $\mathsf{Rec}_{\mathfrak{A}}$ of ***partial $\mathfrak{A}$-recursive functionals*** is the smallest class of functionals over $\mathfrak{A}$ such that

(i) $\mathsf{Expl}_{\mathfrak{A}} \subseteq \mathsf{Rec}_{\mathfrak{A}}$;

(ii) $\mathsf{Rec}_{\mathfrak{A}}$ is closed under substitution;

(iii) $\mathsf{Rec}_{\mathfrak{A}}$ is closed under ***recursion*** — that is, if $I_0, \ldots, I_n \in \mathsf{Rec}_{\mathfrak{A}}$ and $(I_0, \ldots, I_n)$ is iterable, then $I_0^\infty = \mathsf{Rec}(I_0, \ldots, I_n) \in \mathsf{Rec}_{\mathfrak{A}}$.

4.3 Corollary. (i) $\mathsf{Rec}_{\mathfrak{A}}$ *is closed under composition, definition by cases, and argument rearrangement and expansion;*

(ii) *if* $\mathfrak{A}' = (A, \mathsf{Fnl}_{\mathfrak{A}'})$, *and* $\mathsf{Fnl}_{\mathfrak{A}} \subseteq \mathsf{Fnl}_{\mathfrak{A}'} \subseteq \mathsf{Fnl}_{\mathfrak{A}} \cup \mathsf{Rec}_{\mathfrak{A}}$, *then* $\mathsf{Rec}_{\mathfrak{A}} = \mathsf{Rec}_{\mathfrak{A}'}$.

Proof. The first two parts of (i) follow by Remark 2.3; the remainder is proved exactly as for $\mathsf{Expl}_{\mathfrak{A}}$. □

In classical Recursion Theory a vital role is played by various normal forms which provide a relatively simple and uniform representation for all partial recursive functions. For example, for every partial recursive function f, there is a primitive recursive function g such that

$$f(\boldsymbol{m}) \simeq \text{'least' } n\,[g(\boldsymbol{m}, n) = 0].$$

The key point here is that since g is primitive recursive its definition does not involve the search operator, so that the computation of $f(\boldsymbol{m})$ cam be arranged to use only one search. The corresponding normal form here demonstrates that every partial $\mathfrak{A}$-recursive functional can be obtained by a single application of the operator Rec to a finite sequence of $\mathfrak{A}$-explicit functionals.

4.4 Definition. A functional F over A is ***simply partial*** $\mathfrak{A}$***-recursive*** iff for some iterable sequence $(I_0, \ldots, I_n)$ of $\mathfrak{A}$-explicit functionals, $F = I_0^\infty = \mathsf{Rec}(I_0, \ldots, I_n)$.

The normal form may then be stated:

4.5 Theorem. *Every partial* $\mathfrak{A}$*-recursive functional is simply partial* $\mathfrak{A}$*-recursive*

Proof. We need to establish three facts:

(i) every $\mathfrak{A}$-explicit functional is simply partial $\mathfrak{A}$-recursive;

(ii) the class of simply partial $\mathfrak{A}$-recursive functionals is closed under substitution;

(iii) the class of simply partial $\mathfrak{A}$-recursive functionals is closed under recursion.

The first of these is immediate: if I is $\mathfrak{A}$-explicit of type $(k; \boldsymbol{l})$, let ${}^*I(x, i) \simeq I(x)$, with ${}^*I$ of type $(k; \boldsymbol{l}, 0)$. Clearly ${}^*I^\infty = I$ and is simply partial $\mathfrak{A}$-recursive.

The proofs of parts (ii) and (iii) are somewhat tedious and are sketched in the Appendix. □

We have also the following alternative normal form.

4.6 Theorem. *A functional* F *is partial* $\mathfrak{A}$*-recursive iff for some* $\mathfrak{A}$*-explicit functional* I *and some finite sequence* $\boldsymbol{e}$ *of the distinguished elements* $0_{\mathfrak{A}}$ *and* $1_{\mathfrak{A}}$,

$$F(\boldsymbol{a}, \boldsymbol{f}) \simeq I^\infty(\boldsymbol{a}, \boldsymbol{e}, \boldsymbol{f}).$$

Proof. Since the (constant functions with values) $\mathsf{0}_{\mathfrak{A}}$ and $\mathsf{1}_{\mathfrak{A}}$ are $\mathfrak{A}$-explicit, any such functional is partial $\mathfrak{A}$-recursive. For the converse, assume that F is partial $\mathfrak{A}$-recursive; by Theorem 4.5, there exists an iterable sequence $(I_0, \ldots, I_n)$ of $\mathfrak{A}$-explicit functionals such that $F = I_0^\infty = \mathsf{Rec}(I_0, \ldots, I_n)$. Each I_p $(p \leq n)$ has type of the form $(k_p + l; \boldsymbol{l}, k_0, \ldots, k_n)$. Choose $k > \max\{\, k_p : p \leq n \,\}$ and distinct sequences $\boldsymbol{e}^0, \ldots, \boldsymbol{e}^n$ of the elements $\mathsf{0}_{\mathfrak{A}}$ and $\mathsf{1}_{\mathfrak{A}}$ such that each $\boldsymbol{e}_p$ has length $k - k_p$. Let $\boldsymbol{a}^p$ denote a sequence of length k_p. Then there is an $\mathfrak{A}$-explicit functional J such that for $p \leq n$,

$$J(\boldsymbol{a}^p, \boldsymbol{e}^p, x, j) \simeq I_p(\boldsymbol{a}^p, x, j^0, \ldots, j^n),$$

where

$$j^p = \lambda\, \boldsymbol{a}^p\, .\, j(\boldsymbol{a}^p, \boldsymbol{e}^p).$$

Now it is straightforward to prove by induction that

$$J^\sigma(\boldsymbol{a}^p, \boldsymbol{e}^p, x) \simeq I_p^\sigma(\boldsymbol{a}^p, x)$$

so in particular

$$F(\boldsymbol{a}^0, x) \simeq I_0^\infty(\boldsymbol{a}^0, x) \simeq J^\infty(\boldsymbol{a}^0, \boldsymbol{e}^0, x)$$

as desired. □

We conclude this section by indicating two ways to assign natural numbers as indices to the members of $\mathsf{Rec}_{\mathfrak{A}}$. First we need to assign indices to the $\mathfrak{A}$-explicit functionals. As usual in such situations, we actually do the reverse by assigning functionals to all natural numbers regarded as codes for programs.

4.7 Definition. For and functional structure $\mathfrak{A}$ and each $b \in \omega$, we define recursively a functional $[b]^{\mathfrak{A}}$ as follows, where $\boldsymbol{l} = l_0, \ldots, l_{m-1}$:

(i) $[\langle\, 0, \langle\, k, \boldsymbol{l}\,\rangle, i\,\rangle]^{\mathfrak{A}} = \mathsf{Pr}_i^{k,\boldsymbol{l}} \quad (i < k)$;

(ii) $[\langle\, 1, \langle\, k, \boldsymbol{l}\,\rangle, j\,\rangle]^{\mathfrak{A}} = \mathsf{Ev}_j^{k,\boldsymbol{l}} \quad (j < m, l_j = k)$;

(iii) $[\langle\, 2, \langle\, k, \boldsymbol{l}\,\rangle, i\,\rangle]^{\mathfrak{A}} =$ the i-th member of $\mathsf{Fnl}_{\mathfrak{A}}$;

(iv) $[\langle\, 3, \langle\, k, \boldsymbol{l}\,\rangle, c, d_0, \ldots, d_{m-1}\,\rangle]^{\mathfrak{A}} =$ the composition of $[c]^{\mathfrak{A}}$ with $[d_0]^{\mathfrak{A}}, \ldots, [d_{m-1}]^{\mathfrak{A}}$;

(v) $[\langle\, 4, \langle\, k, \boldsymbol{l}\,\rangle, c, d_0, \ldots, d_{m-1}, e\,\rangle]^{\mathfrak{A}} =$ the substitution of $[c]^{\mathfrak{A}}$ with $[d_0]^{\mathfrak{A}}, \ldots, [d_{m-1}]^{\mathfrak{A}}$;

(vi) $[\langle\, 5, \langle\, k+1, \boldsymbol{l}\,\rangle, c, d, e\,\rangle]^{\mathfrak{A}} =$ the functional defined by cases from $[c]^{\mathfrak{A}}$, $[d]^{\mathfrak{A}}$, and $[e]^{\mathfrak{A}}$.

In (v), the number e codes information to describe the parameter lists for the substituted functionals. In each case the assigned functional is completely undefined (the empty functional) if the ranks of the component parts do not match. For all other $b \in \omega$, $[b]^{\mathfrak{A}}$ is the empty functional.

4.8 Proposition. *The class of functionals $[b]^{\mathfrak{A}}$ for $b \in \omega$ coincides with the class of $\mathfrak{A}$-explicit functionals.* □

Now we can use either one of the normal forms to assign indices to all $\mathfrak{A}$-partial recursive functionals:

4.9 Definition. For all natural number values of the parameters,

$$\{\langle 6, b_0, \ldots, b_n \rangle\}^{\mathfrak{A}}_{\mathsf{Rec}} = [b_0]^{\mathfrak{A},\infty} = \mathsf{Rec}([b_0]^{\mathfrak{A}}, \ldots, [b_n]^{\mathfrak{A}}),$$

and

$$\{\langle 7, b, i_0, \ldots, i_{n-1} \rangle\}^{\mathfrak{A}}_{\mathsf{Rec}}(\boldsymbol{a}, \boldsymbol{f}) \simeq [b]^{\mathfrak{A},\infty}(\boldsymbol{a}, e_0, \ldots, e_{n-1}, \boldsymbol{f}),$$

where for $j < n$, either $i_j = 0$ and $e_j = 0_{\mathfrak{A}}$ or $i_j = 1$ and $e_j = 1_{\mathfrak{A}}$. As above, in all other cases, $\{b\}^{\mathfrak{A}}_{\mathsf{Rec}}$ is the empty functional.

It then follows immediately that

4.10 Proposition. *Each of the classes of functionals $\{b\}^{\mathfrak{A}}_{\mathsf{Rec}}$ with $(b)_0 = 6$ or with $(b)_0 = 7$ coincides with the class of partial $\mathfrak{A}$-recursive functionals.* □

Note that each of these indexings has the property that the type of the functional can be effectively recovered from the index.

We conclude this section with

4.11 Theorem. *For any first-order structure $\mathfrak{A}$, $\mathsf{Reg}_{\mathfrak{A}} \subseteq \mathsf{Rec}_{\mathfrak{A}}$ — that is, every $\mathfrak{A}$-register computable functional is $\mathfrak{A}$-partial recursive*

Proof. Let $\pi = (I_0, \ldots, I_m)$ be a fixed $\mathrm{URM}_{\mathfrak{A}}$-program which computes a functional F. Let $R_0, \ldots, R_k$ include all registers mentioned in π. Each non-conditional instruction I_p can be regarded as an $\mathfrak{A}$-explicit function which maps the sequence $r_0, \ldots, r_k$ of contents of these registers to another such sequence, which we denote by $I_p(r_0, \ldots, r_k)$. Let $J_0, \ldots, J_n$ be the following iterable sequence of functionals: if I_p is non-conditional, then

$$J_p(r_0, \ldots, r_k, \boldsymbol{f}, j_0, \ldots, j_k) \simeq \begin{cases} j_{p+1}(I_p(r_0, \ldots, r_k)), & \text{if } p < m; \\ r_0, & \text{otherwise;} \end{cases}$$

and if I_p is the instruction "if $R(r_{q_0}, \ldots, r_{q_{k-1}})$, go to I_s, else go to I_t", then

$$J_p(r_0, \ldots, r_k, \boldsymbol{f}, j_0, \ldots, j_k) \simeq \begin{cases} j_s(r_0, \ldots, r_k), & \text{if } R(r_{q_0}, \ldots, r_{q_{k-1}}) \text{ and } s \leq m; \\ j_t(r_0, \ldots, r_k), & \text{if not } R(r_{q_0}, \ldots, r_{q_{k-1}}) \text{ and } t \leq m; \\ r_0, & \text{otherwise.} \end{cases}$$

It follows easily that for each $p \leq m$, $J^{\infty}_p(r_0, \ldots, r_k, \boldsymbol{f})$ is the halting value of r_0 (if any) obtained by starting at instruction I_p with register contents $r_0, \ldots, r_k$ and function arguments $\boldsymbol{f}$. Hence

$$F(\boldsymbol{a}, \boldsymbol{f}) \simeq J^{\infty}_0(\boldsymbol{a}, \boldsymbol{e}),$$

where $\boldsymbol{e}$ is the sequence of $0_{\mathfrak{A}}$'s of the appropriate length. It follows that F is partial $\mathfrak{A}$-recursive. □

5. The Main Examples

Of course the first example is classical recursion over the natural numbers. We saw in Corollary 3.1 that the Ω-register computable functionals coincide with the ordinary partial recursive functionals. We have also

5.1 Theorem. *The class* Rec_Ω *of partial* Ω*-recursive functionals coincides with the usual class of partial recursive functionals over* ω*.*

Proof. We take as our characterization of the partial recursive functionals the one of [Hi; II.3.3]: the smallest class which contains the primitive recursive functionals and is closed under composition and unbounded search.The functionals there take only total functions as arguments, but the characterization extends easily to the more general case considered here. This class surely contains the Ω-explicit functionals and is closed under substitution (see, for example, [Hi; II.3.9] and the discussion preceding it). Furthermore, by an easy extension of Kleene's First Recursion Theorem (see [Hi; II.4.31]) this class is closed under recursion in the sense of 4.2 (iii). Hence every partial Ω-recursive functional is partial recursive.

For the opposite inclusion, it will suffice to show that the class of Ω-recursive functionals is closed under primitive recursion and search. By Remark 2.3, this is equivalent to showing that the two particular functionals Se_ω and Prim_ω are partial Ω-recursive. For Se_ω, let

$$I(p,g,i) \simeq \begin{cases} 0, & \text{if } g(p) \simeq 0; \\ \mathsf{Sc}(i(\mathsf{Sc}(p))), & \text{if } g(p) \downarrow \neq 0. \end{cases}$$

It is straightforward to prove by induction on m that

$$I^m(p,g) \simeq n \qquad \Longleftrightarrow \qquad n \leq m \quad \text{and} \quad g(p+n) \simeq 0 \quad \text{and} \quad (\forall q < n) g(p+q) \downarrow \neq 0.$$

Hence,

$$I^\infty(p,g) \simeq n \qquad \Longleftrightarrow \qquad g(p+n) \simeq 0 \quad \text{and} \quad (\forall q < n) g(p+q) \downarrow \neq 0,$$

and $\mathsf{Se}_\omega(g) \simeq I^\infty(0,g)$.

Towards Prim_ω, we note first that the predecessor function on ω is Ω-recursive:

$$\mathsf{Pred}(m) \simeq \begin{cases} 0, & \text{if } m = 0; \\ \text{`least' } n[\mathsf{Sc}(n) = m], & \text{otherwise.} \end{cases}$$

Now if we set

$$I(p,n,h,i) \simeq \begin{cases} n, & \text{if } p = 0; \\ h(i(\mathsf{Pred}(p), n)), & \text{otherwise;} \end{cases}$$

then it is straightforward to prove that $\mathsf{Prim}_\omega = I^\infty$. $\square$

We turn next to the class of partial κ-recursive functionals for an admissible ordinal κ. There is no full treatment of computable functionals on ordinals in print, so we shall

need to be somewhat more detailed here. We start with the characterization of the partial κ-recursive functions of [Hi; VIII.1.3]. There we define a set $\Omega_{\kappa,\kappa}$ of finite sequences of (natural numbers and) ordinals and set

$$\{a\}_\kappa(\boldsymbol{\mu}) \simeq \nu \quad \Longleftrightarrow \quad (a, \boldsymbol{\mu}, \nu) \in \Omega_{\kappa,\kappa}.$$

The set $\Omega_{\kappa,\kappa}$ is inductively defined as the smallest set containing certain tuples $(a, \boldsymbol{\mu}, \nu)$ which represent some simple initial functions and is closed under clauses corresponding to composition, indexing, and the functionals Sup_κ and Se_κ. The latter, for example, takes the form

$$\begin{aligned}\nu < \kappa \quad \text{and} \quad (b, \nu, \boldsymbol{\mu}, 0) \in \Omega_{\kappa,\kappa} \quad \text{and} \quad (\forall \pi < \nu)(\exists \xi > 0)[(b, \nu, \boldsymbol{\mu}, \xi) \in \Omega_{\kappa,\kappa}] \\ \Longrightarrow (\langle\, 4, k, b \,\rangle, \boldsymbol{\mu}, \nu) \in \Omega_{\kappa,\kappa}.\end{aligned}$$

It is straightforward to extend this definition to partial κ-recursive functionals. All that is required is to extend the coding of the indices to record the types of functionals and the addition of a clause for evaluating the function arguments, say

$$(\langle\, 0, k, \boldsymbol{l}, 3, j \,\rangle, \boldsymbol{\mu}, \boldsymbol{f}, f_j(\boldsymbol{\mu})) \in \Omega_{\kappa,\kappa}.$$

Now the development of [Hi; VIII.1.3–13] can be repeated essentially verbatim for the partial κ-recursive functionals.

The new results which must be established are closure of the class of partial κ-recursive functionals under substitution and recursion (in the sense of 4.2 (iii)). Closure under substitution can be established by the method of [Hi; VI.2] — one shows that there is a primitive recursive function ι such that

$$\{\iota(a, \boldsymbol{c})\}_\kappa(\boldsymbol{\mu}, \boldsymbol{f}) \simeq \{b\}(\boldsymbol{\mu}, \lambda\, \boldsymbol{\nu}^0 \,.\, \{c_0\}(\boldsymbol{\mu}, \boldsymbol{\nu}^0, \boldsymbol{f}), \ldots, \lambda\, \boldsymbol{\nu}^{m-1} \,.\, \{c_{m-1}\}(\boldsymbol{\mu}, \boldsymbol{\nu}^{m-1}, \boldsymbol{f})).$$

Note that because function arguments here are partial, none of the problems of totality which plague [Hi; VI.2] arise here.

Towards closure under recursion, let I be an iterable partial κ-recursive functional and let G be the partial κ-recursive functional defined by:

$$\begin{aligned}G(e, 0, \sigma, \boldsymbol{\mu}, x) &\simeq \{e\}_\kappa(1, \tau^*, \boldsymbol{\mu}, x) \\ G(e, 1, \sigma, \boldsymbol{\mu}, x) &\simeq I(\boldsymbol{\mu}, x, \lambda\, \boldsymbol{\mu}' \,.\, \{e\}_\kappa(0, \sigma, \boldsymbol{\mu}', x)),\end{aligned}$$

where using the selection function Sel_κ of [Hi; VIII.2.9] $\tau^* \simeq \tau^*(e, \sigma, \boldsymbol{\mu}, x)$ is a κ-partial recursive functional such that

$$(\exists \tau < \sigma)\{e\}_\kappa(1, \tau, \boldsymbol{\mu}, x) \downarrow \quad \Longrightarrow \quad \tau^* < \sigma \text{ and } \{e\}_\kappa(1, \tau^*, \boldsymbol{\mu}, x) \downarrow .$$

By the Second Recursion Theorem, choose an index $\bar{e}$ such that

$$\{\bar{e}\}_\kappa(i, \sigma, \boldsymbol{\mu}, x) \simeq G(\bar{e}, i, \sigma, \boldsymbol{\mu}, x).$$

Then it is straightforward to prove by induction on σ that

$$\{\bar{e}\}_\kappa(0,\sigma,\boldsymbol{\mu},x) \simeq I^{<\sigma}(\boldsymbol{\mu},x)$$
$$\{\bar{e}\}_\kappa(1,\sigma,\boldsymbol{\mu},x) \simeq I^{\sigma}(\boldsymbol{\mu},x),$$

so that if, again using Sel_κ, $\sigma^* \simeq \sigma^*(\boldsymbol{\mu},x)$ is κ-partial recursive such that $\{\bar{e}\}_\kappa(1,\sigma^*,\boldsymbol{\mu},x)\downarrow$ whenever any such $\sigma<\kappa$ exists, then

$$I^{<\kappa}(\boldsymbol{\mu},x) \simeq \{\bar{e}\}_\kappa(1,\sigma^*,\boldsymbol{\mu},x)$$

is partial κ-recursive. Now, if there are no partial function arguments and κ is recursively regular, it follows as in [Hi; VIII.2.5] that the closure ordinal of I is at most κ, so that $I^\infty = I^{<\kappa}$ and thus also I^∞ is partial κ-recursive. The extension of this argument to an iterable sequence $(I_0,\ldots,I_n)$ of partial κ-recursive functionals is straightforward.

Let $\mathfrak{A}_\kappa$ denote $(\kappa,\mathsf{Sc},0,\mathsf{Sup}_\kappa,\mathsf{Se}_\kappa)$.

5.2 Theorem. *For any admissible ordinal κ, the partial $\mathfrak{A}_\kappa$-recursive functions coincide with the usual partial κ-recursive functions.*

Proof. We have essentially shown in the preceding paragraphs that every partial $\mathfrak{A}_\kappa$-recursive function is partial κ-recursive. For the opposite inclusion, we can verify that the universal partial κ-recursive function

$$U(a,\langle\boldsymbol{\mu}\rangle) \simeq \{a\}_\kappa(\boldsymbol{\mu})$$

is partial $\mathfrak{A}_\kappa$-recursive. In fact, U is the least fixed point of a single functional I, with $I(a,\langle\boldsymbol{\mu}\rangle,i)$ defined by cases on a. If a is an index corresponding to an initial function, $I(a,\langle\boldsymbol{\mu}\rangle,i)$ is defined outright. If $a=\langle 3,k+1,b\rangle$ is an index for an application of the search operator Se_κ, then we take

$$I(a,\langle\boldsymbol{\mu}\rangle,i) \simeq \mathsf{Se}_\kappa(\lambda\,\pi\,.\,i(b,\langle\pi,\boldsymbol{\mu}\rangle)).$$

The other cases are handled similarly and it is straightforward to verify that $U=I^\infty$ as required. □

We consider next positive elementary (first-order) inductive definition over a first-order structure $\mathfrak{A}$ as developed in [Mo1]. In accord with the conventions of that book, we assume that $\mathsf{Dis}_{\mathfrak{A}} = A$ — that is, every element of A is distinguished. Note that this conflicts with our earlier restriction to finite languages, but the only place we have made use of this is in assigning indices. Let $\mathfrak{A}^+$ be the functional structure $(\mathfrak{A},=_A,\mathsf{E}^\sharp_A)$. Roughly, the correspondence is

$$\mathfrak{A}\text{-positive elementary} = \mathfrak{A}^+\text{-explicit}$$
$$\mathfrak{A}\text{-inductive} = \mathfrak{A}^+\text{-semi-recursive}$$
$$\mathfrak{A}\text{-hyperelementary} = \mathfrak{A}^+\text{-recursive},$$

but a little preparation is needed to make this precise. Let L be the usual first-order language associated with $\mathfrak{A}$ and L^+ the language obtained from L by adjoining a countable sequence of relation variables. We use letters p, q for the relation symbols of L and L^+, t (with scripts) for terms, u, v, w for individual variables, and U, V, W for relation variables. The terms of L^+ are just those of L and the formulas of each language are as usual with the stipulation that the relation variables occur positively. Thus the L^+-formulas comprise the smallest class containing all atomic and negated atomic formulas of L, formulas $V(t_0, \ldots, t_n)$, and closed under $\wedge$, $\vee$, $\exists v$ and $\forall v$.

For any l-ary partial function f over A, let Gr_f denote the usual $(l+1)$-ary relation called the ***graph*** of f. If F is a $(k;\boldsymbol{l})$-ary functional over A, Gr_F is the second-order relation such that

$$F(\boldsymbol{a}, \boldsymbol{f}) \simeq e \quad \Longleftrightarrow \quad \mathsf{Gr}_F(\boldsymbol{a}, e, \mathsf{Gr}_{\boldsymbol{f}}).$$

Here, $\mathsf{Gr}_{\boldsymbol{f}} = \mathsf{Gr}_{f_0,\ldots,f_{m-1}} = (\mathsf{Gr}_{f_0}, \ldots, \mathsf{Gr}_{f_{m-1}})$. If P is a relation on A, the ***semi-characteristic function*** of P is

$$\chi_P^+(\boldsymbol{a}) \simeq \begin{cases} 1_{\mathfrak{A}}, & \text{if } P(\boldsymbol{a}); \\ \uparrow, & \text{otherwise.} \end{cases}$$

We write $\chi^+_{\boldsymbol{P}}$ for $\chi^+_{P_0}, \ldots, \chi^+_{P_{l-1}}$.

5.3 Proposition. (i) *For any $\mathfrak{A}^+$-explicit functional F, there exists an L^+-formula ϕ_F which defines Gr_F over $\mathfrak{A}$ — that is,*

$$F(\boldsymbol{a}, \boldsymbol{f}) \simeq e \quad \Longleftrightarrow \quad \models_{\mathfrak{A}} \phi_F[\boldsymbol{a}, e, \mathsf{Gr}_{\boldsymbol{f}}];$$

(ii) *for any L^+-formula ϕ, there exists an $\mathfrak{A}^+$-explicit functional F_ϕ such that*

$$\models_{\mathfrak{A}} \phi[\boldsymbol{a}, \boldsymbol{P}] \quad \Longleftrightarrow \quad F_\phi(\boldsymbol{a}, \chi^+_{\boldsymbol{P}}) \simeq 1_{\mathfrak{A}};$$

and in case ϕ has no relation variables, also

$$\not\models_{\mathfrak{A}} \phi[\boldsymbol{a}] \quad \Longleftrightarrow \quad F_\phi(\boldsymbol{a}) \simeq 0_{\mathfrak{A}}.$$

Proof. The proof of (i) is by induction over the $\mathfrak{A}^+$-explicit functionals and is relatively straightforward. For example, suppose

$$F(\boldsymbol{a}, f) \simeq G(H(\boldsymbol{a}, f), f).$$

Then from ϕ_G and ϕ_H, we define $\phi_F(\boldsymbol{u}, v, W)$ as

$$\exists w\, [\phi_H(\boldsymbol{u}, w, W) \wedge \phi_G(w, v, W)].$$

$\phi_{\mathsf{E}^\sharp_A}(w, V)$ is the formula

$$[w \doteq \mathsf{1}_{\mathfrak{A}} \wedge \exists v\, V(v, \mathsf{1}_{\mathfrak{A}})] \vee [w \doteq \mathsf{0}_{\mathfrak{A}} \wedge \forall v\, V(v, \mathsf{0}_{\mathfrak{A}})].$$

To see that the class of F for which ϕ_F exists is closed under substitution, suppose

$$F(\boldsymbol{a}, \boldsymbol{f}) \simeq G(\boldsymbol{a}, \lambda \boldsymbol{b}\,.\, H(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{f})) \quad \text{and} \quad \phi_G \text{ and } \phi_H \text{ exist.}$$

Then we may take for ϕ_F the result of replacing each subformula of the form $W(\boldsymbol{s}, \boldsymbol{t}, u)$ of ϕ_G by $\phi_H(\boldsymbol{s}, \boldsymbol{t}, u)$ (changing bound variables if necessary to avoid collisions).

For (ii), we proceed by induction on L^+-formulas. First note that for any L-term t, there is an $\mathfrak{A}$-explicit function F_t such that

$$\models_{\mathfrak{A}} t[\boldsymbol{a}] \doteq e \quad \Longleftrightarrow \quad F_t(\boldsymbol{a}) = e.$$

If ϕ is an atomic formula of L, say $p(\boldsymbol{t})$, then set

$$F_\phi(\boldsymbol{a}) \simeq \chi_{p^{\mathfrak{A}}}(F_{\boldsymbol{t}}(\boldsymbol{a})).$$

(Technically, we account for differing variables in the terms $t_0, \ldots, t_{k-1}$.) Because of the explicit inclusion of the equality on A in the structure $\mathfrak{A}^+$, this works also for atomic equations. If ϕ is $V(\boldsymbol{t})$, then

$$F_\phi(\boldsymbol{a}, f) \simeq \begin{cases} \mathsf{1}_{\mathfrak{A}}, & \text{if } f(F_{\boldsymbol{t}}(\boldsymbol{a})) \simeq \mathsf{1}_{\mathfrak{A}}; \\ \uparrow, & \text{otherwise.} \end{cases}$$

Now we set recursively,

$$\begin{aligned} F_{\phi\wedge\psi}(x) &\simeq \mathsf{And}(F_\phi(x), F_\psi(x)); \\ F_{\phi\vee\psi}(x) &\simeq \mathsf{E}^\sharp_A(\lambda b\,.\, G(b, x)); \end{aligned}$$

where

$$G(b, x) \simeq \begin{cases} F_\phi(x), & \text{if } b = \mathsf{0}_{\mathfrak{A}} \\ F_\psi(x), & \text{otherwise;} \end{cases}$$

$$\begin{aligned} F_{\exists v\,\phi}(x) &\simeq \mathsf{E}^\sharp_A(\lambda b\,.\, F_\phi(b, x)); \\ F_{\forall v\,\phi}(x) &\simeq \mathsf{Not}(\mathsf{E}^\sharp_A(\lambda b\,.\, \mathsf{Not}(F_\phi(b, x)))). \quad \square \end{aligned}$$

To complete the picture, we need to recall from [Mo1] and [Mo2] the definitions of the $\mathfrak{A}$-inductive and $\mathfrak{A}$-hyperelementary relations. With any L^+-formula $\phi(v_0, \ldots, v_{k-1}, V)$, with V a k-ary relation variable, we associate a mapping $\Gamma_\phi : \wp(A^k) \to \wp(A^k)$ defined by

$$\Gamma_\phi(R) = \Big\{ \boldsymbol{a} \in A^k : \models_{\mathfrak{A}} \phi[\boldsymbol{a}, R] \Big\}.$$

Because V occurs positively in ϕ, Γ_ϕ is monotone in R and hence by arguments very similar to those of Section 2 has a least fixed point Γ_ϕ^∞ — that is, $\Gamma_\phi(\Gamma_\phi^\infty) = \Gamma_\phi^\infty$ and for any R such that $\Gamma_\phi(R) = R$, $\Gamma_\phi^\infty \subseteq R$. Similarly, for a finite sequence of formulas $\phi_i(v_0, \ldots, v_{k_i-1}, V_0, \ldots, V_n) \quad (i \leq n)$, we have mappings

$$\Gamma_{\phi_i}(R_0, \ldots, R_n) = \Big\{ \boldsymbol{a} \in A^{k_i} : \models_{\mathfrak{A}} \phi_i[\boldsymbol{a}, R_0, \ldots, R_n] \Big\}$$

which have a simultaneous least fixed point $(\Gamma_{\phi_0}^\infty, \ldots, \Gamma_{\phi_n}^\infty)$. A relation $R \subseteq A^k$ is $\mathfrak{A}$***-inductive*** iff for some such sequence, $R = \Gamma_{\phi_0}^\infty := \mathsf{Ind}(\phi_0, \ldots, \phi_n)$, and $\mathfrak{A}$***-hyperelementary*** iff both R and $A^k \sim R$ are $\mathfrak{A}$-inductive. These definitions can also be relativized to individual and relational parameters, but to simplify the presentation here we leave this generalization to the reader.

These definitions are slightly different from those of [Mo1;1D]. There, R is called ***inductive on*** $\mathfrak{A}$ iff for some formula $\phi(\boldsymbol{v}, \boldsymbol{w}, V)$ and some $\boldsymbol{b} \in A^l$,

$$R(\boldsymbol{a}) \quad \Longleftrightarrow \quad (\boldsymbol{a}, \boldsymbol{b}) \in \Gamma_\phi^\infty.$$

Using the techniques of the proof of Theorem 4.6 it is easy to show that these two notions are the same, at least for infinite structures. The key point is that here every $a \in A$ is a distinguished element and the equality relation is definable. In Theorem 4.6 we used the special distinguished elements $\mathsf{0}_{\mathfrak{A}}$ and $\mathsf{1}_{\mathfrak{A}}$ for which the predicates $\cdot = \mathsf{0}_{\mathfrak{A}}$ and $\cdot = \mathsf{1}_{\mathfrak{A}}$ are by definition $\mathfrak{A}$-recursive (because they are $\mathfrak{A}$-explicit) even if equality on A is not in general $\mathfrak{A}$-recursive.

5.4 Proposition. (i) *For any partial $\mathfrak{A}^+$-recursive function f, Gr_f is $\mathfrak{A}$-inductive; hence every $\mathfrak{A}^+$-semi-recursive relation is $\mathfrak{A}$-inductive;*

(ii) *every $\mathfrak{A}$-inductive relation is $\mathfrak{A}^+$-semi-recursive.*

Proof. Let f be partial $\mathfrak{A}^+$-recursive; then by Theorem 4.5 there exists an iterable sequence $(I_0, \ldots, I_n)$ of $\mathfrak{A}^+$-explicit functionals such that $f = I_0^\infty = \mathsf{Rec}(I_0, \ldots, I_n)$. For each $p \leq n$, let $\phi_p = \phi_{I_p}$ be an L^+-formula as in the preceding proposition such that

$$I_p(a_0, \ldots, a_{k_p-1}, i_0, \ldots, i_n) \simeq e \quad \Longleftrightarrow \quad \models_{\mathfrak{A}} \phi_p[a_0, \ldots, a_{k_p-1}, e, \mathsf{Gr}_{i_0}, \ldots \mathsf{Gr}_{i_n}].$$

Then with notation as above, by induction on σ, $\mathsf{Gr}_{I_p^\sigma} = \Gamma_{\phi_p}^\sigma$ so that $\mathsf{Gr}_{I_p^\infty} = \Gamma_{\phi_p}^\infty$ and thus in particular, $\mathsf{Gr}_f = \mathsf{Gr}_{I_0^\infty}$ is $\mathfrak{A}$-inductive. Now if R is any $\mathfrak{A}^+$-semi-recursive relation, then for some $(I_0, \ldots, I_n)$ as above,

$$R = \mathsf{Dom}\ I_0^\infty = \big\{ \boldsymbol{a} : \exists e\, [(\boldsymbol{a}, e) \in \Gamma_{\phi_0}^\infty] \big\},$$

which by [Mo1;1D1] is also $\mathfrak{A}$-inductive.

For (ii), we use similarly the second half of the preceding proposition. For any iterable

sequence $(\phi_0, \ldots, \phi_n)$ of L^+-formulas with no parameters let $I_p = F_{\phi_p}$ be $\mathfrak{A}^+$-explicit functionals such that

$$\models_{\mathfrak{A}} \phi_p[\boldsymbol{a}^p, R_0, \ldots, R_n] \quad \Longleftrightarrow \quad I_p(\boldsymbol{a}^p, \chi^+_{R_0}, \ldots, \chi^+_{R_n}) \simeq 1_{\mathfrak{A}}.$$

Then by induction on σ, we have $I_p^\sigma = \chi^+_{\Gamma^\sigma_{\phi_p}}$ so that $I_p^\infty = \chi^+_{\Gamma^\infty_{\phi_p}}$. In particular, $\Gamma^\infty_{\phi_0} = \mathsf{Dom}\ I_0^\infty$ and is therefore $\mathfrak{A}^+$-semi-recursive. □

Another very important example, which motivated much of the development of this approach to abstract recursion, is recursion on the finite type structure over a functional structure. This example is fully developed in [Ke-Mo], to which we refer the interested reader.

Our final example is the computation theory over ordered commutative rings with unit introduced in [BSS]. It would take us too far afield to lay out here the entire formalism of this theory, which is quite different from ours, so we shall restrict ourselves to a general description. Computation of a BSS-***machine*** is carried out on a ***state space*** $\overline{S}$ and defines a partial map from an ***input space*** $\overline{I}$ to an ***output space*** $\overline{O}$. Each of these spaces is some finite cartesian power of the domain R of the underlying ring. (The theory of [BSS] embraces also infinite dimensional input, output, and state spaces, but we shall not discuss this aspect.) The machine M acts by generating a path through a finite directed graph. Each node of the graph is either a (unique) input node, an output node, a computation node, or a branch node. With each time step s is associated a ***state*** $\boldsymbol{r}^s = (r_0^s, \ldots, r_n^s) \in \overline{S}$; $\boldsymbol{r}^0$ is the input. With each computation node is associated a sequence $f_0, \ldots, f_n$ of polynomials with coefficients from R, and the effect of the node is to alter the state as follows:

$$(r_0, \ldots, r_n) \mapsto (f_0(r_0, \ldots, r_n), \ldots, f_n(r_0, \ldots, r_n)),$$

and to point to a unique next node. With a branch node is associated a polynomial g and two potential next nodes. A branch node does not alter the state but chooses the next node depending as $g(r_0, \ldots, r_n) < 0$ or not.

A BSS-machine can be converted in a straightforward way to a register machine. A state $(r_0, \ldots, r_n)$ corresponds to the contents of registers $R_0, \ldots, R_n$. The action of a computation node corresponds to a finite sequence of instructions $r_i := f_i(r_0, \ldots, r_n)$. This, in turn, can be expanded to a longer finite sequence of instructions involving only the addition and multiplication operators of the ring and the constants of the polynomials f_i. Similarly, the action of a branch node corresponds to a conditional instruction "if $r_q < 0$, go to I_s, else go to I_t" preceded by a sequence of instructions with the effect $r_q := g(r_0, \ldots, r_n)$. The resulting program governs a register machine over the structure

$$\mathcal{R}_{a_0,\ldots,a_{n-1}} = (R, +, \times, <, =_R, 0, 1, a_0, \ldots, a_{n-1}),$$

where $a_0, \ldots, a_{n-1}$ are all of the constant coefficients of any of the polynomials of the BSS-machine. Conversely, any $\mathcal{R}_{a_0,\ldots,a_{n-1}}$-register machine is (essentially) a BSS-machine. Thus we have

5.5 Theorem. *For any ordered ring $(R, +, \times, <, =_R, 0, 1)$, a function $f : R^k \to R$ is BSS-computable iff for some $a_0, \ldots, a_{n-1} \in R$, f is $\mathcal{R}_{a_0,\ldots,a_{n-1}}$-register computable.* □

6. Structures with arithmetic

A basic feature of most intuitive notions of computability is that at some level the algorithms are specified in a finite way. This is reflected by the existence of an indexing of the computable partial functionals by natural numbers. In ordinary recursion theory over ω, such an indexing results in names for functions which are elements of the underlying domain and hence in a large group of results such as the existence of universal computable functions, the Second Recursion Theorem, etc. To extend this part of the theory to an abstract structure $\mathfrak{A}$, we will need to require that $\mathfrak{A}$ contain a copy of the natural numbers.

6.1 Definition. A functional [first-order] structure is a ***structure with arithmetic*** iff $\mathfrak{A}$ includes a unary relation (set) $\omega_{\mathfrak{A}} \subseteq A$, a unary (successor) function $\mathsf{Sc}_{\mathfrak{A}}$, and a binary relation $=_{\omega\,\mathfrak{A}}$ such that the structure $(\omega_{\mathfrak{A}}, \mathsf{Sc}_{\mathfrak{A}}, =_{\omega\,\mathfrak{A}}, 0_{\mathfrak{A}})$ is isomorphic to the standard structure $(\omega, \mathsf{Sc}, =, 0)$.

For simplicity of notation, we shall assume that the two structures $(\omega_{\mathfrak{A}}, \mathsf{Sc}_{\mathfrak{A}}, =_{\omega\,\mathfrak{A}}, 0_{\mathfrak{A}})$ and $(\omega, \mathsf{Sc}, =, 0)$ are actually identical; this can always be arranged by replacing $\mathfrak{A}$ by an isomorphic copy with domain including ω.

Many of the examples of the preceding section satisfy this condition, including the structures $\mathfrak{A}_\kappa$ and $\mathcal{R}_{a_0,\ldots,a_{n-1}}$.

6.2 Proposition. *For any functional [first-order] structure $\mathfrak{A}$ with arithmetic, $\mathsf{Rec}_{\mathfrak{A}}$ [$\mathsf{Reg}_{\mathfrak{A}}$] includes all of the ordinary partial recursive functionals.*

Proof. For $\mathsf{Rec}_{\mathfrak{A}}$ this follows from Theorem 5.1 and for $\mathsf{Reg}_{\mathfrak{A}}$ from Corollary 3.1. □

For structures with arithmetic it makes sense to ask if the ***universal*** functionals

$$U^{\mathfrak{A},k,l}_{\mathsf{Reg}}(b, \boldsymbol{a}, \boldsymbol{f}) \simeq \{b\}^{\mathfrak{A}}_{\mathsf{Reg}}(\boldsymbol{a}, \boldsymbol{f}) \quad \text{and}$$
$$U^{\mathfrak{A},k,l}_{\mathsf{Rec}}(b, \boldsymbol{a}, \boldsymbol{f}) \simeq \{b\}^{\mathfrak{A}}_{\mathsf{Rec}}(\boldsymbol{a}, \boldsymbol{f})$$

are computable in the relevant sense. For $\mathsf{Reg}_{\mathfrak{A}}$ (over first-order structures) this turns out to be true for functions (without function arguments) under mild additional hypotheses. For $\mathsf{Rec}_{\mathfrak{A}}$, there are two conditions under which universal functionals are partial $\mathfrak{A}$-recursive. The first concerns functional structures $\mathfrak{A}$ for which there is an $\mathfrak{A}$-recursive coding of finite sequences of elements of A; this case is treated thoroughly in [Ke-Mo; Section 8]. We consider here first-order structures $\mathfrak{A}$, for which the functionals $U^{\mathfrak{A},k,l}_{\mathsf{Rec}}$ are partial $\mathfrak{A}$-recursive without restriction. The proofs are less straightforward than one might expect; in both cases the difficulty lies in verifying that certain aspects of computations are bounded.

Consider first register computations. A program π for a universal function must effectively simulate all programs π_b. But π must work within some fixed number of registers, while the programs that it is simulating will generally use an unbounded number of registers. In the case of ordinary register machines on ω, this difficulty is overcome by the

use of sequence coding; the information in any finite number of registers can be coded into a single number and stored in a single register. In particular, the whole configuration of a URM after execution of s steps of program π_b with input $\boldsymbol{m}$ can be coded as a single number. Thus a simulating machine needs only to calculate the sequence of these configuration codes as s increases and read off the answer when a halting state is reached. In slightly more detail, following [Cu; Section 5.1], let

$$\sigma_k(b, \boldsymbol{m}, s) = \langle\, \langle\, r_0^s, \ldots, r_{n_b}^s \,\rangle, p^s \,\rangle,$$

where r_i^s denotes the contents of the i-th register after s steps of the computation of π_b with input $\boldsymbol{m}$, and p^s denotes the number of the instruction to be executed at the $s+1$-st step. It is easy to verify that the functions σ_k are primitive recursive and that

$$U^k_{\mathsf{Reg}}(b, \boldsymbol{m}) \simeq r_0^s \text{ for the least } s \text{ such that } p^s \text{ is not a valid instruction number}$$

is partial recursive and hence register computable.

For a URM $\mathfrak{A}$ this approach is not directly available, since we cannot in general code the contents of finitely many registers containing elements of A as a single element of A. The solution to this problem has been discovered at least twice independently in [MS-HT2; Section 3] and [Mi1] and lies in the observation that for any computation without function arguments the contents of the registers of a URM $\mathfrak{A}$ can be described in a finitary way as the interpretations of terms of the first-order language of $\mathfrak{A}$ evaluated at the input points. We may therefore describe the configuration of a URM $\mathfrak{A}$ by a number coding the sequence of Gödel numbers of these terms and a simulating machine may manipulate these configuration codes as before. For example, suppose that at step s of the computation of a program π acting on input $\boldsymbol{a}$, the instruction to be executed is $r_p := f(r_{q_0}, \ldots, r_{q_{k-1}})$ and the configuration description contains the information that the contents of each r_{q_i} is the value $t_i^{\mathfrak{A}}[\boldsymbol{a}]$ of the term t_i. Then the configuration description at step $s+1$ will be changed to show the contents of r_p to be the value $f(t_0 \ldots t_{k-1})^{\mathfrak{A}}[\boldsymbol{a}]$ of this compound term. We set

$$\sigma_k^{\mathfrak{A}}(b, \boldsymbol{a}, s) = \langle\, \langle\, \mathsf{gn}(t_0^s), \ldots, \mathsf{gn}(t_{n_b}^s) \,\rangle, p^s \,\rangle,$$

where t_i^s denotes the term such that after s steps of the computation of π_b with input $\boldsymbol{a}$, R_i contains $t_i^{s,\mathfrak{A}}[\boldsymbol{a}]$ and gn denotes any standard Gödel numbering.

This scheme encounters one difficulty in verifying that the functions $\sigma_k^{\mathfrak{A}}$ are register computable: at the execution of a conditional instruction we need the actual values of the register contents in order to decide which branch to follow. Thus, we need to to be able to compute $t^{\mathfrak{A}}[\boldsymbol{a}]$ from the Gödel number $\mathsf{gn}(t)$ of the term t. The same sort of computation suffices at the end of the simulation to produce the value of the computed function.

6.3 Definition. A first-order structure $\mathfrak{A}$ ***admits*** URM $\mathfrak{A}$ ***term evaluation*** iff for each $k \in \omega$ there exists $F_k \in \mathsf{Reg}_{\mathfrak{A}}$ such that for all terms t containing at most the variables $v_0, \ldots, v_{k-1}$ and all $\boldsymbol{a} \in A^k$, $F_k(\mathsf{gn}(t), \boldsymbol{a}) = t^{\mathfrak{A}}[\boldsymbol{a}]$.

6.4 Theorem. *For any first-order structure* $\mathfrak{A}$*, if* $\mathfrak{A}$ *admits* $\mathrm{URM}_{\mathfrak{A}}$ *term evaluation, then for each* $k \in \omega$ *there exists a universal* $\mathsf{Reg}_{\mathfrak{A}}$*-computable function*

$$U^{\mathfrak{A},k}_{\mathsf{Reg}}(b, \boldsymbol{a}) \simeq \{b\}^{\mathfrak{A}}_{\mathsf{Reg}}(\boldsymbol{a}). \quad \square$$

Consider next the existence of functionals $U^{\mathfrak{A},k,\boldsymbol{l}}_{\mathsf{Rec}}$ universal for the functionals in $\mathsf{Rec}_{\mathfrak{A}}$. The basic idea for constructing these functionals is relatively simple, although the details are messy. Consider first universal functionals $U^{\mathfrak{A},k,\boldsymbol{l}}_{\mathsf{Expl}}$ for $\mathsf{Expl}_{\mathfrak{A}}$. The natural approach is to define this as a fixed point I^{∞} for an $\mathfrak{A}$-explicit functional I which mirrors the inductive definition of $\mathsf{Expl}_{\mathfrak{A}}$— for example, corresponding to composition would be a clause

$$I(\langle 3, \langle k, \boldsymbol{l} \rangle, c, d_0, \ldots, d_{m-1} \rangle, \boldsymbol{a}, \boldsymbol{f}, i) \simeq i(c, i(d_0, \boldsymbol{a}), \ldots, i(d_{m-1}, \boldsymbol{a})),$$

but we see immediately that this is not a legal application of recursion, since i is applied to argument lists of different lengths $k+1$ and $m+1$.

If we assume that $\mathfrak{A}$ admits $\mathfrak{A}$-recursive coding of finite sequences, this problem can be overcome by converting all functions to unary ones using this coding; this is the approach of [Ke-Mo] and works for functional as well as first-order structures. Without this assumption we can still deal with first-order structures as follows. With each $F \in \mathsf{Expl}_{\mathfrak{A}}$ we can associate a finite ***derivation*** $F_0, \ldots, F_{n-1}, F$ such that each term of this sequence is either an initial functional or arises from some of the preceding terms via one of the closure principles. Suppose for a moment that every functional of type $(k; \boldsymbol{l})$ has such a derivation with each F_i of some type $(m; \boldsymbol{l})$ with $m \leq k$. Then the clause above could be written

$$I(\langle 3, \langle k, \boldsymbol{l} \rangle, c, d_0, \ldots, d_{m-1} \rangle, \boldsymbol{a}, \boldsymbol{f}, i) \simeq i(c, i(d_0, \boldsymbol{a}), \ldots, i(d_{m-1}, \boldsymbol{a}), \mathsf{0}_{\mathfrak{A}}, \ldots, \mathsf{0}_{\mathfrak{A}}),$$

with $(k-m)$-many $\mathsf{0}_{\mathfrak{A}}$'s. In other words, we could exhibit a functional universal for functionals of all types $(m; \boldsymbol{l})$ with $m \leq k$. Of course, this condition will not generally be satisfied for two reasons: some of the functionals in the signature of $\mathfrak{A}$ may have types other than these and the composition and substitution schemas do not preserve types. For first-order structures, we can get around these difficulties as follows.

6.5 Definition. For any structure $\mathfrak{A}$, a functional $F \in \mathsf{Expl}_{\mathfrak{A}}$ of type $(k; \boldsymbol{l})$ is ***conservative*** iff it has a derivation $F_0, \ldots, F_{n-1}, F$ such that each F_i is of type $(m; \boldsymbol{l})$ or $(m;)$ with $m \leq k$, and the only uses of the substitution schema are trivial ones of the form $F(\boldsymbol{a}, \boldsymbol{f}) \simeq g(\boldsymbol{a})$, for g a given function of $\mathfrak{A}$.

6.6 Proposition. *For any first-order structure* $\mathfrak{A}$*, any* $\boldsymbol{l}$*, and all sufficiently large* k*, every* $F \in \mathsf{Expl}_{\mathfrak{A}}$ *of type* $(k; \boldsymbol{l})$ *is conservative.*

Proof. (Sketch) Define a functional F to be ***strongly conservative*** iff F is conservative and every functional which results from composing or substituting F with conservative

functionals is also conservative. Then it is straightforward (if messy) to prove by induction that for k larger than all of the l_j and all of the ranks of the functions and relations of $\mathfrak{A}$, every $F \in \mathsf{Expl}_{\mathfrak{A}}$ of type $(k; \boldsymbol{l})$ is strongly conservative. □

Now for sufficiently large k it is easy to define along the lines suggested above an $\mathfrak{A}$-explicit functional I such that I^∞ is universal for $\mathfrak{A}$-explicit functionals of types $(m; \boldsymbol{l})$ for all $m \leq k$; shorter lists are "padded" with $\mathsf{0}_{\mathfrak{A}}$'s, and we have

6.7 Theorem. *For any first-order structure $\mathfrak{A}$ with arithmetic, for each k and $\boldsymbol{l}$, the functional $U^{\mathfrak{A},k,\boldsymbol{l}}_{\mathsf{Rec}}$ universal for partial $\mathfrak{A}$-recursive functionals of type $(k; \boldsymbol{l})$ is partial $\mathfrak{A}$-recursive.*

Proof. Since for $m \leq k$, any m-ary function can be regarded also as a k-ary function, it suffices to prove the result for large k. Let I be as in the preceding paragraph and set

$$J(b, \boldsymbol{a}, \boldsymbol{f}, j) \simeq \begin{cases} I^\infty(b, \boldsymbol{a}, \boldsymbol{f}, j), & \text{if } (b)_1 = \langle\, k + l; \boldsymbol{l}, k \,\rangle; \\ \mathsf{0}_{\mathfrak{A}}, & \text{otherwise.} \end{cases}$$

then by an easy induction, for all ordinals σ and iterable $[b]$,

$$J^\sigma(b, \boldsymbol{a}, \boldsymbol{f}) \simeq [b]^\sigma(\boldsymbol{a}, \boldsymbol{f}),$$

whence J^∞ is universal for fixed points of $\mathfrak{A}$-explicit functionals. It follows immediately that $U^{\mathfrak{A},k,\boldsymbol{l}}_{\mathsf{Rec}}$ is partial $\mathfrak{A}$-recursive. □

We conclude this section by showing that for all first-order structures with arithmetic which admit $\mathrm{URM}_{\mathfrak{A}}$ term evaluation, a function is $\mathfrak{A}$-register computable iff it is $\mathfrak{A}$-partial recursive. By Proposition 3.2 and Theorem 4.11, it suffices to show that $\mathsf{Reg}_{\mathfrak{A}}$ is closed under recursion. The outline of the proof is quite similar to a standard proof of the First Recursion Theorem in ordinary recursion theory, for example, [Cu; 10.3]. First we have

6.8 Proposition. *For any first-order structure $\mathfrak{A}$ with arithmetic and any $\mathfrak{A}$-register computable function F, there exists a $\mathfrak{A}$-register computable relation R such that*

$$F(\boldsymbol{a}) \downarrow \quad \Longleftrightarrow \quad (\exists s \in \omega) R(s, \boldsymbol{a}).$$

Proof. We define $R(s, \boldsymbol{a})$ to hold just in case $F(\boldsymbol{a})$ is defined by a computation which executes at most s many instructions. A $\mathrm{URM}_{\mathfrak{A}}$ program for the characteristic function of R is easily constructed from one for F by inserting after each instruction a routine which increments a counter, compares it with the stored value of s, and either continues with the computation of F if the counter value is less than s or halts with the value $\mathsf{0}_{\mathfrak{A}}$. If the computation terminates because $F(\boldsymbol{a})$ is defined, the value 1 is produced as output. □

6.9 Number Selection Theorem. *For any first-order structure $\mathfrak{A}$ with arithmetic and any $\mathfrak{A}$-register computable function F, there exists a $\mathfrak{A}$-register computable function Sel_F such that*

$$(\exists n \in \omega) F(n, \boldsymbol{a}) \downarrow \quad \Longrightarrow \quad \mathsf{Sel}_F(\boldsymbol{a}) \downarrow \quad \text{and} \quad F(\mathsf{Sel}_F(\boldsymbol{a}), a) \downarrow .$$

Proof. With R as in the proposition, we take simply

$$\mathsf{Sel}_F(\boldsymbol{a}) \simeq (\text{least } p \,.\, R((p)_0, (p)_1, \boldsymbol{a}))_1. \ \square$$

6.10 Proposition. *For any first-order structure $\mathfrak{A}$ with arithmetic which admits $\mathrm{URM}_{\mathfrak{A}}$ term evaluation, and any iterable $\mathfrak{A}$-register computable functional I, also I^∞ is $\mathfrak{A}$-register computable.*

Proof. Let b be a register index for I. By Proposition 3.3, there exists a primitive recursive function h such that for any index c,

$$\{b\}^{\mathfrak{A}}_{\mathsf{Reg}}(\boldsymbol{a}, [c]_{\mathfrak{A}}) \simeq \{h(b,c)\}^{\mathfrak{A}}_{\mathsf{Reg}}(\boldsymbol{a}).$$

Hence if $f(0)$ is an index for a totally undefined function and $f(n+1) = h(b, f(n))$, then $f(n)$ is an index for I^n, and by Theorem 6.4 the function

$$F(n, \boldsymbol{a}) \simeq \{f(n)\}^{\mathfrak{A}}_{\mathsf{Reg}}(\boldsymbol{a}) \simeq I^n(\boldsymbol{a})$$

is $\mathfrak{A}$-register computable. By Propositions 2.6 and 4.1. $I^\infty = I^{<\omega}$, so

$$I^\infty(\boldsymbol{a}) \downarrow \quad \Longrightarrow \quad (\exists n \in \omega) F(n, \boldsymbol{a}) \downarrow \quad \Longrightarrow \quad \mathsf{Sel}_F(\boldsymbol{a}) \downarrow \quad \text{and} \quad I^{\mathsf{Sel}_F(\boldsymbol{a})}(\boldsymbol{a}) \downarrow .$$

Thus $I^\infty(\boldsymbol{a}) \simeq F(\mathsf{Sel}_F(\boldsymbol{a}), \boldsymbol{a}))$ and is thus $\mathfrak{A}$-register computable. $\square$

6.11 Corollary. *For any first-order structure $\mathfrak{A}$ with arithmetic which admits $\mathrm{URM}_{\mathfrak{A}}$ term evaluation, a function is $\mathfrak{A}$-register computable iff it is partial $\mathfrak{A}$-recursive.*

Proof. By the preceding proposition and Theorem 4.11. $\square$

The ordered rings of [BSS] discussed at the end of Section 5 are examples of structures which satisfy the hypothesis of this corollary. It is easy to see that term evaluation — which is simply polynomial evaluation — is register computable over any ring. Roughly speaking, each monomial may be computed by accumulating partial products in a single register and the whole polynomial similarly computed by accumulating partial sums. The number of registers needed for scratch-work does not depend on the complexity of the polynomial. For details, see [BSS; Section 8]. For an example of a structure $\mathfrak{A}$ which does not admit $\mathrm{URM}_{\mathfrak{A}}$-term evaluation, see [MS-HT2; Theorem 4.3], and for numerous algebraic examples which satisfy the hypotheses of the Corollary, see [Tu].

7. Sections and Envelopes

Every theory of computability uses functions to characterize two classes of relations.

7.1 Definition. For any relation $R \subseteq A^k$,

(i) R is $\mathfrak{A}$***-register decidable*** ($\mathfrak{A}$***-recursive***) iff its characteristic function χ_R is an $\mathfrak{A}$-register computable ($\mathfrak{A}$-recursive) function. The classes of these relations are called ***sections*** and denoted

$$\mathsf{Sec}_{\mathsf{Reg}}(\mathfrak{A}) = \{\, R : R \text{ is an } \mathfrak{A}\text{-register decidable relation} \,\}$$
$$\mathsf{Sec}_{\mathsf{Rec}}(\mathfrak{A}) = \{\, R : R \text{ is an } \mathfrak{A}\text{-recursive relation} \,\}.$$

(ii) R is $\mathfrak{A}$***-semi-register decidable*** ($\mathfrak{A}$***-semi-recursive***) iff for some partial $\mathfrak{A}$-register computable (partial $\mathfrak{A}$-recursive) function f, $R = \mathsf{Dom}\ f$, the domain of f. The classes of these relations are called ***envelopes*** and denoted

$$\mathsf{Env}_{\mathsf{Reg}}(\mathfrak{A}) = \{\, R : R \text{ is an } \mathfrak{A}\text{-semi-register decidable relation} \,\}$$
$$\mathsf{Env}_{\mathsf{Rec}}(\mathfrak{A}) = \{\, R : R \text{ is an } \mathfrak{A}\text{-semi-recursive relation} \,\}.$$

In the following, when either subscript Rec or Reg applies, the subscript will be omitted.

In classical recursion theory over $\Omega = (\omega, \mathsf{Sc}, 0)$, the first simple observations about the classes of recursive and semi-recursive relations are:

(1) $\mathsf{Sec}(\Omega)$ is a Boolean algebra;
(2) $\mathsf{Sec}(\Omega) \subseteq \mathsf{Env}(\Omega)$;
(3) $\mathsf{Env}(\Omega)$ is closed under the positive Boolean operations;
(4) $R \in \mathsf{Sec}(\Omega)$ iff both R and its complement $\omega^k \sim R$ belong to $\mathsf{Env}(\Omega)$.
(5) $\mathsf{Env}(\Omega) \not\subseteq \mathsf{Sec}(\Omega)$,
(6) $\mathsf{Env}(\Omega)$ is not closed under complementation.

In the general case, some of these results remain trivially true, while others are true only with additional hypotheses and more complicated proofs.

First the easy parts:

7.2 Proposition. *For any functional structure* $\mathfrak{A}$*,*

(i) $\mathsf{Sec}(\mathfrak{A})$ *is a Boolean algebra;*
(ii) $\mathsf{Sec}(\mathfrak{A}) \subseteq \mathsf{Env}(\mathfrak{A})$*;*
(iii) $\mathsf{Env}(\mathfrak{A})$ *is closed under intersection.*

Proof. For (i) it suffices to show that $\mathsf{Sec}(\mathfrak{A})$ is closed under intersection and complement. These follow from the equations

$$\chi_{\sim R}(\boldsymbol{a}) = \begin{cases} 1_{\mathfrak{A}}, & \text{if } \chi_R(\boldsymbol{a}) = 0_{\mathfrak{A}}; \\ 0_{\mathfrak{A}}, & \text{otherwise;} \end{cases}$$

$$\chi_{R\cap S}(\boldsymbol{a}) = \begin{cases} 1_{\mathfrak{A}}, & \text{if } \chi_R(\boldsymbol{a}) = 1_{\mathfrak{A}} \text{ and } \chi_S(\boldsymbol{a}) = 1_{\mathfrak{A}}; \\ 0_{\mathfrak{A}}, & \text{otherwise.} \end{cases}$$

For (ii), if $R \in \mathsf{Sec}(\mathfrak{A})$ and

$$F(\boldsymbol{a}) \simeq \begin{cases} 0_{\mathfrak{A}}, & \text{if } R(\boldsymbol{a}); \\ \uparrow, & \text{otherwise;} \end{cases}$$

then $R = \mathsf{Dom}\ F \in \mathsf{Env}(\mathfrak{A})$. For (iii), note that if $F(\boldsymbol{a}) \simeq \mathsf{Pr}^2_0(G(\boldsymbol{a}), H(\boldsymbol{a}))$, then $\mathsf{Dom}\ F = \mathsf{Dom}\ G \cap \mathsf{Dom}\ H$. □

7.3 Proposition. *For any of the cases discussed above for which there exist universal partial computable functions,*

(i) $\mathsf{Env}(\mathfrak{A}) \not\subseteq \mathsf{Sec}(\mathfrak{A})$,

(ii) $\mathsf{Env}(\mathfrak{A})$ *is not closed under complementation.*

Proof. By the standard diagonal argument. □

The easiest example that shows that (4) does not hold in general is the following. Let $A \subseteq \omega$ be a non-recursive recursively enumerable set and let f and g be the semi-characterictic functions of A, $\omega \sim A$, respectively:

$$f(m) \simeq \begin{cases} 1, & \text{if } m \in A; \\ \uparrow, & \text{otherwise;} \end{cases} \quad \text{and} \quad g(m) = \begin{cases} 1, & \text{if } m \notin A; \\ \uparrow, & \text{otherwise;} \end{cases}$$

and $\Omega_{f,g} = (\omega, \mathsf{Sc}, 0, f, g)$. Clearly both A and $\omega \sim A$ belong to $\mathsf{Env}(\Omega_{f,g})$, but we claim that $\mathsf{Sec}(\Omega_{f,g}) = \mathsf{Sec}(\Omega)$, so that $A \notin \mathsf{Sec}(\Omega_{f,g})$. To see this, note that for any monotone functional K over ω,

$$K(x, f, g) \simeq n \quad \Longrightarrow \quad K(\lambda m\,.\,1, \lambda m\,.\,1) \simeq n.$$

Hence, if $\lambda x\,.\,K(x, f, g)$ is total, then it coincides with $\lambda x\,.\,K(x, \lambda m\,.\,1, \lambda m\,.\,1)$. It is easy to prove by induction that

$$F \text{ is partial } \Omega_{f,g}\text{-recursive} \quad \Longrightarrow \quad F(x) \simeq K(x, f, g) \text{ for some partial } \Omega\text{-recursive } K.$$

Hence if F is (total) $\Omega_{f,g}$-recursive, then F is also Ω-recursive. In particular, this holds when F is the characteristic function of $R \in \mathsf{Sec}(\Omega_{f,g})$.

The key to establishing (3) (closure under union) and (4) is some kind of selection theorem. From the results of the preceding section we have

7.4 Theorem. *For any first-order structure* $\mathfrak{A}$ *with arithmetic,*

(i) $\mathsf{Env}_{\mathsf{Reg}}(\mathfrak{A})$ *is closed under union;*

(ii) $R \in \mathsf{Sec}_{\mathsf{Reg}}(\mathfrak{A})$ *iff both* R *and its complement* $A^k \sim R$ *belong to* $\mathsf{Env}_{\mathsf{Reg}}(\mathfrak{A})$.

Proof. Let $R = \mathsf{Dom}\ f$ and $S = \mathsf{Dom}\ g$ be $\mathfrak{A}$- semi-register-decidable and define

$$F(n, \boldsymbol{a}) \simeq \begin{cases} f(\boldsymbol{a}), & \text{if } n = 0; \\ g(\boldsymbol{a}), & \text{if } n > 0. \end{cases}$$

Then (using Theorem 6.9) $(R \cup S)(\boldsymbol{a}) \iff (\exists n \in \omega) F(n, \boldsymbol{a}) \downarrow \iff \mathsf{Sel}_F(\boldsymbol{a}) \downarrow$, so $R \cup S$ is also $\mathfrak{A}$-semi-register-decidable. Similarly, for (ii), if $R = \mathsf{Dom}\ f$ and $\sim R = \mathsf{Dom}\ g$ and we set

$$G(n, \boldsymbol{a}) \simeq \begin{cases} f(\boldsymbol{a}), & \text{if } n = 1; \\ g(\boldsymbol{a}), & \text{if } n = 0, \end{cases}$$

then Sel_G is $\mathfrak{A}$-register computable and is the characteristic function of R, so R is $\mathfrak{A}$-register decidable. □

One of the main results of [BSS] (Section 4, Proposition 2), together with [Mi2], is a characterization of the BSS-semi-computable relations over the real numbers as the countable unions of semi-algebraic sets whose definitions require only finitely many non-rational parameters. One direction of this characterization is a general fact about register computability, which we sketch briefly. Fix a first-order structure $\mathfrak{A}$ and a program $\pi = (I_0, \ldots, I_n)$ for an $\mathfrak{A}$-register machine. For each finite sequence $\boldsymbol{i} = (i_0, \ldots, i_s)$, let

$$V^k_{\boldsymbol{i}} = \left\{ \boldsymbol{a} \in A^k : \pi(\boldsymbol{a}) \downarrow \text{ via the computation path } I_{i_0}, \ldots, I_{i_s} \right\}.$$

By an analysis similar to that of Section 6, with each p and $\boldsymbol{i}$ we may associate a fixed term $t_{p,\boldsymbol{i}}$ such that a computation with input $\boldsymbol{a}$ which follows the computation path $I_{i_0}, \ldots, I_{i_s}$ leaves register R_p containing the value $t^{\mathfrak{A}}_{p,\boldsymbol{i}}[\boldsymbol{a}]$ of this term evaluated at $\boldsymbol{a}$. If I_{i_j} is a conditional instruction with hypothesis $R(r_{q_0}, \ldots, r_{q_{k-1}})$, then whether or not a computation which has reached I_{i_j} continues to $I_{i_{j+1}}$ depends on whether or not

$$R^{\mathfrak{A}}(t^{\mathfrak{A}}_{q_0, i_0, \ldots, i_j}[\boldsymbol{a}], \ldots, t^{\mathfrak{A}}_{q_{k-1}, i_0, \ldots, i_j}[\boldsymbol{a}])$$

holds — in other words, whether the atomic formula $Rt_{q_0, i_0, \ldots, i_j} \ldots t_{q_{k-1}, i_0, \ldots, i_j}$ is satisfied by $\boldsymbol{a}$. It follows that for each k and $\boldsymbol{i}$ there exists a formula $\phi^k_{\boldsymbol{i}}$ which is a conjunction of atomic and negated atomic formulas such that

$$V^k_{\boldsymbol{i}} = \left\{ \boldsymbol{a} \in A^k : \models_{\mathfrak{A}} \phi^k_{\boldsymbol{i}}[\boldsymbol{a}] \right\}.$$

Then, since the domain of π is the union of such sets over all possible paths, we have

7.5 Theorem. *For any first-order structure $\mathfrak{A}$, every $\mathfrak{A}$-semi-register-decidable relation is a countable union of relations quantifier-free definable over $\mathfrak{A}$.* □

In the case of ordered rings, where the only relations are $<$ and $=$, this reduces to [BSS; Section 4, Proposition 2] that every semi-computable relation is a countable union of basic semi-algebraic sets — that is, sets defined by finitely many polynomial inequalities of the types $g(\boldsymbol{a}) < 0$ and $h(\boldsymbol{a}) \leq 0$. This fact is used in [BSS; Section 1] to derive that various natural sets of real numbers, such as the complements of Cantor sets, are semi-computable but not computable over the ring of real numbers. For this special case there is also a converse to the Theorem due to C. Michaux:

7.6 Theorem. [Mi2] *A relation P is BSS-semi-computable over the ring of real numbers if and only if for some reals $a_0, \ldots, a_n$, P is a countable union of relations quantifier-free definable over $(\mathbb{R}, +, \times, <, =_{\mathbb{R}}, 0, 1, a_0, \ldots, a_{n-1})$.*

The proof is based on the fact that the countable amount of information required to specify such a union can be coded as a single real number.

For the $\mathfrak{A}$-recursive case, the key idea, introduced in [Ke-Mo] is normality.

7.7 Definition. A functional F is ***$\mathfrak{A}$-normal*** iff there exists a partial $\mathfrak{A}$-recursive ***normalizing functional*** Δ_F as follows. For any partial function δ, set

$$Z_\delta = \{\, \boldsymbol{b} : \delta(\boldsymbol{b}) \simeq 0_{\mathfrak{A}} \,\}.$$

Then, (when F has only one partial function argument)

(i) $F(\boldsymbol{a}, f \restriction Z_\delta)\downarrow \implies \Delta_F(\boldsymbol{a}, f, \delta) \simeq 1_{\mathfrak{A}}$;

(ii) δ is total, $Z_\delta \subseteq \mathsf{Dom}\ f$, and $F(\boldsymbol{a}, f \restriction Z_\delta)\uparrow \implies \Delta_F(\boldsymbol{a}, f, \delta) \simeq 0_{\mathfrak{A}}$.

The intuition behind this definition is that in case (i), if $f \restriction Z_\delta$ is enough of f for $F(\boldsymbol{a}, f \restriction Z_\delta)$ to be defined, this can be verified by delaying the exercise of an evaluation of a value $f(\boldsymbol{b})$ during the computation until we verify that $\delta(\boldsymbol{b}) \simeq 0$. In case (ii), we cannot expect in general to be able to verify that $f \restriction Z_\delta$ is *not* enough of f for $F(\boldsymbol{a}, f \restriction Z_\delta)$ to be defined, since δ itself my contain too little information; the extra hypotheses rule this out.

The extension of the definition to functionals with several partial function arguments is straightforward, for example if F is of type $(k; l, m)$, then Δ_F is of type $(k; l, m, l, m)$.

7.8 Examples. (i) If F has no partial function arguments, then Δ_F is just the characteristic function of $\mathsf{Dom}\ F$, so F is $\mathfrak{A}$-normal iff $\mathsf{Dom}\ F$ is $\mathfrak{A}$-recursive. In particular, if F is total, then F is $\mathfrak{A}$-normal.

(ii) the evaluation functionals are $\mathfrak{A}$-normal — for example, let

$$\Delta_{\mathsf{Ev}_j^{k,l}}(\boldsymbol{a}, f, \delta) \simeq \begin{cases} 1_{\mathfrak{A}}, & \text{if } \delta(\boldsymbol{a}) \simeq 0_{\mathfrak{A}}; \\ 0_{\mathfrak{A}}, & \text{if } \delta(\boldsymbol{a})\downarrow \neq 0_{\mathfrak{A}}. \end{cases}$$

It is easily verified that this is a normalizing functional.

(iii) The composition functionals are normal — for example, for any γ and δ, we have

$$\begin{aligned}\mathsf{Cmp}^{k;1}(\boldsymbol{a}, g \restriction Z_\gamma, h \restriction Z_\delta) \simeq (g \restriction Z_\gamma)((h \restriction Z_\delta)(\boldsymbol{a})) \downarrow \\ \iff \quad \delta(\boldsymbol{a}) \simeq \mathsf{0}_{\mathfrak{A}} \text{ and } \gamma(h(\boldsymbol{a})) \simeq \mathsf{0}_{\mathfrak{A}} \text{ and } g(h(\boldsymbol{a})) \downarrow .\end{aligned}$$

Set

$$\Delta_{\mathsf{Cmp}^{k;1}}(\boldsymbol{a}, g, h, \gamma, \delta) \simeq \begin{cases} \mathsf{IsZero}_{\mathfrak{A}}(\gamma(h(\boldsymbol{a}))), & \text{if } \delta(\boldsymbol{a}) \simeq \mathsf{0}_{\mathfrak{A}}; \\ \mathsf{0}_{\mathfrak{A}}, & \text{if } \delta(\boldsymbol{a}) \downarrow \not\simeq \mathsf{0}_{\mathfrak{A}}. \end{cases}$$

If $(g \restriction Z_\gamma)((h \restriction Z_\delta)(\boldsymbol{a})) \downarrow$, then clearly $\Delta_{\mathsf{Cmp}^{k;1}}(\boldsymbol{a}, g, h, \gamma, \delta) \simeq 1$ by virtue of the first clause. Suppose now that $(g \restriction Z_\gamma)((h \restriction Z_\delta)(\boldsymbol{a})) \uparrow$, and the other conditions of (ii) of the definition are satisfied. If $\delta(\boldsymbol{a}) \not\simeq \mathsf{0}_{\mathfrak{A}}$, then since δ is total, $\Delta_{\mathsf{Cmp}^{k;1}}(\boldsymbol{a}, g, h, \gamma, \delta) \simeq \mathsf{0}_{\mathfrak{A}}$ by virtue of the second clause. Otherwise, $\delta(\boldsymbol{a}) \simeq \mathsf{0}_{\mathfrak{A}}$, so $\boldsymbol{a} \in Z_\delta \subseteq \mathsf{Dom}\ h$ and since γ is total, $\gamma(h(\boldsymbol{a})) \downarrow$. If $\gamma(h(\boldsymbol{a})) \simeq \mathsf{0}_{\mathfrak{A}}$, then $h(\boldsymbol{a}) \in Z_\gamma \subseteq \mathsf{Dom}\ g$, so $(g \restriction Z_\gamma)((h \restriction Z_\delta)(\boldsymbol{a})) \simeq g(h(\boldsymbol{a})) \downarrow$, contrary to assumption. Hence $\gamma(h(\boldsymbol{a})) \downarrow \not\simeq \mathsf{0}_{\mathfrak{A}}$ and $\Delta_{\mathsf{Cmp}^{k;1}}(\boldsymbol{a}, g, h, \gamma, \delta) \simeq \mathsf{0}_{\mathfrak{A}}$ by the first clause.

7.9 Definition. A functional structure $\mathfrak{A}$ is ***normal*** iff every $F \in \mathsf{Fnl}_{\mathfrak{A}}$ is $\mathfrak{A}$-normal.

7.10 Corollary. *Every first-order structure is normal.* □

7.11 Proposition. *If $\mathfrak{A}$ is normal, then all $\mathfrak{A}$-explicit functionals are $\mathfrak{A}$-normal.*

Proof. From the examples above and by Remark 2.3, it remains only to show that each Cases^k is $\mathfrak{A}$-normal and that the class of $\mathfrak{A}$-normal functionals is closed under substitution. The first of these is immediate:

$$\Delta_{\mathsf{Cases}^k}(c, \boldsymbol{a}, f, g, h, \gamma, \delta, \epsilon) \simeq \begin{cases} \mathsf{IsZero}_{\mathfrak{A}}(\gamma(\boldsymbol{a})), & \text{if } c = \mathsf{1}_{\mathfrak{A}}; \\ \mathsf{IsZero}_{\mathfrak{A}}(\delta(\boldsymbol{a})), & \text{if } c = \mathsf{0}_{\mathfrak{A}}; \\ \mathsf{IsZero}_{\mathfrak{A}}(\epsilon(\boldsymbol{a})), & \text{otherwise} \end{cases}$$

is easily seem to be a normalizing functional.

For substitution we prove a special case. Suppose G and H are $\mathfrak{A}$-normal and

$$F(\boldsymbol{a}, f) \simeq G(\boldsymbol{a}, H_{\boldsymbol{a},f}),$$

where, as usual, $H_{\boldsymbol{a},f} = \lambda \boldsymbol{b}\, .\, H(\boldsymbol{a}, \boldsymbol{b}, f)$. Set (abusing notation)

$$\delta'(\boldsymbol{b}) \simeq \mathsf{IsZero}_{\mathfrak{A}}(\Delta_H(\boldsymbol{a}, \boldsymbol{b}, f, \delta)) \quad \text{and} \quad \Delta_F(\boldsymbol{a}, f, \delta) \simeq \Delta_G(\boldsymbol{a}, H_{\boldsymbol{a},f}, \delta').$$

To see that Δ_F is a normalizing functional for F, suppose first that $F(\boldsymbol{a}, f \restriction Z_\delta) \downarrow$ and set

$$\gamma(\boldsymbol{b}) \simeq 0 \quad \iff \quad H(\boldsymbol{a}, \boldsymbol{b}, f \restriction Z_\delta) \downarrow .$$

Then $G(\boldsymbol{a}, H_{\boldsymbol{a},f} \restriction Z_\gamma) \downarrow$ and $Z_\gamma \subseteq Z_{\delta'}$, so by monotonicity, also $G(\boldsymbol{a}, H_{\boldsymbol{a},f} \restriction Z_{\delta'}) \downarrow$, from which follows that $\Delta_F(\boldsymbol{a}, f, \delta) \simeq 1$ as desired.

If, on the other hand, δ is total, $Z_\delta \subseteq \mathsf{Dom}\ f$ and $F(\boldsymbol{a}, f \restriction Z_\delta)\uparrow$, then also $G(\boldsymbol{a}, H_{\boldsymbol{a},f} \restriction Z_\gamma)\uparrow$. In this case, $Z_\gamma = Z_{\delta'}$, so also $G(\boldsymbol{a}, H_{\boldsymbol{a},f} \restriction Z_{\delta'})\uparrow$. By the hypotheses, $Z_{\delta'} \subseteq \mathsf{Dom}\ H_{\boldsymbol{a},f}$, so we have $\Delta_F(\boldsymbol{a}, f, \delta) \simeq 0$ as desired. $\square$

For any iterable sequence $\mathcal{I} = (I_0, \ldots, I_m)$, the ***$\mathcal{I}$-norm*** is the function $|\cdot|_{\mathcal{I}}$ defined by

$$|x|_{\mathcal{I}} = \begin{cases} \text{least } \sigma\,[I_0^\sigma(x)\downarrow], & \text{if } I_0^\infty(x)\downarrow; \\ \infty, & \text{otherwise;} \end{cases}$$

where ∞ denotes any fixed large ordinal, say $\mathsf{Card}(A)^+$. Thus $|x|_{\mathcal{I}}$ measures "how long" it takes to compute $I_0^\infty(x)$. For two iterable sequences $\mathcal{I}$ and $\mathcal{J}$, set

$$\begin{aligned} x \leq_{\mathcal{I},\mathcal{J}} y \quad &\iff \quad I_0^\infty(x)\downarrow \quad \text{and} \quad |x|_{\mathcal{I}} \leq |y|_{\mathcal{J}} \\ y <_{\mathcal{I},\mathcal{J}} x \quad &\iff \quad J_0^\infty(y)\downarrow \quad \text{and} \quad |y|_{\mathcal{J}} < |x|_{\mathcal{I}}. \end{aligned}$$

Note that the condition '$J_0^\infty(y)\downarrow$' is redundant. Let

$$\mathsf{C}_{\mathcal{I},\mathcal{J}}(x,y) \simeq \begin{cases} \mathsf{0}_{\mathfrak{A}}, & \text{if } x \leq_{\mathcal{I},\mathcal{J}} y \\ \mathsf{1}_{\mathfrak{A}}, & \text{if } y <_{\mathcal{I},\mathcal{J}} x. \end{cases}$$

$\mathsf{C}_{\mathcal{I},\mathcal{J}}(x,y)$ is called the ***stage comparison functional*** for $\mathcal{I}$ and $\mathcal{J}$.

7.12 Stage Comparison Theorem. *[Ke-Mo] If $\mathfrak{A}$ is normal, then for any iterable sequences $\mathcal{I}$ and $\mathcal{J}$ of $\mathfrak{A}$-explicit functionals, the stage comparison functional $\mathsf{C}_{\mathcal{I},\mathcal{J}}$ is partial $\mathfrak{A}$-recursive.*

Proof. We prove the special, but typical, case $\mathcal{I} = (I)$ and $\mathcal{J} = (J)$. Let Δ_I and Δ_J be normalizing functionals for I and J. Let

$$\begin{aligned} G(\boldsymbol{a}, \boldsymbol{b}, g, h) &\simeq \Delta_I(\boldsymbol{a}, I^\infty, \lambda\, \boldsymbol{c}\,.\, h(\boldsymbol{c}, \boldsymbol{b})); \\ H(\boldsymbol{a}, \boldsymbol{b}, g, h) &\simeq \Delta_J(\boldsymbol{a}, J^\infty, \lambda\, \boldsymbol{d}\,.\, g(\boldsymbol{a}, \boldsymbol{d})). \end{aligned}$$

This is an iterable pair of partial $\mathfrak{A}$-recursive functionals; we prove by induction on σ that

$$(1) \qquad \boldsymbol{a} \leq_{IJ} \boldsymbol{b} \wedge |\boldsymbol{a}|_I = \sigma \quad \Longrightarrow \quad G^\infty(\boldsymbol{a}, \boldsymbol{b}) \simeq \mathsf{1}_{\mathfrak{A}}$$

$$(2) \qquad \boldsymbol{b} <_{IJ} \boldsymbol{a} \wedge |\boldsymbol{b}|_J = \sigma \quad \Longrightarrow \quad G^\infty(\boldsymbol{a}, \boldsymbol{b}) \simeq \mathsf{0}_{\mathfrak{A}}$$

$$(3) \qquad \boldsymbol{b} \leq_{JI} \boldsymbol{a} \wedge |\boldsymbol{b}|_J = \sigma \quad \Longrightarrow \quad H^\infty(\boldsymbol{a}, \boldsymbol{b}) \simeq \mathsf{1}_{\mathfrak{A}}$$

$$(4) \qquad \boldsymbol{a} <_{JI} \boldsymbol{b} \wedge |\boldsymbol{a}|_I = \sigma \quad \Longrightarrow \quad H^\infty(\boldsymbol{a}, \boldsymbol{b}) \simeq \mathsf{0}_{\mathfrak{A}}\,.$$

Assume as induction hypothesis that (1)–(4) hold for all $\tau < \sigma$ and assume first the hypothesis of (1) for a fixed $\boldsymbol{a}$ and $\boldsymbol{b}$. Then $I(\boldsymbol{a}, I^{<\sigma}) \simeq I^\sigma(\boldsymbol{a})\downarrow$, and by (4) of the induction hypothesis

$$|\boldsymbol{c}|_I < \sigma \quad \Longrightarrow \quad |\boldsymbol{c}|_I < |\boldsymbol{b}|_J \quad \Longrightarrow \quad H^\infty(\boldsymbol{a}, \boldsymbol{b}) \simeq \mathsf{0}_{\mathfrak{A}}\,.$$

In other words,

$$I^{<\sigma} \subseteq I^\infty \upharpoonright Z_{\lambda\, \boldsymbol{c}\,.\, H^\infty(\boldsymbol{c},\boldsymbol{b})},$$

whence also

$$I(\boldsymbol{a}, I^\infty \upharpoonright Z_{\lambda\, \boldsymbol{c}\,.\, H^\infty(\boldsymbol{c},\boldsymbol{b})}) \downarrow,$$

so by the definition of Δ_I,

$$G^\infty(\boldsymbol{a},\boldsymbol{b}) \simeq G(\boldsymbol{a},\boldsymbol{b},G^\infty,H^\infty) \simeq \Delta_I(\boldsymbol{a}, I^\infty, \lambda\, \boldsymbol{c}\,.\, H^\infty(\boldsymbol{c},\boldsymbol{b})) \simeq 1_{\mathfrak{A}},$$

as desired.

We may prove (3) in a parallel fashion. Now assume the hypotheses of (2). From these, (4) of the induction hypothesis, and the just established (3), it follows that $\lambda\, \boldsymbol{c}\,.\, H^\infty(\boldsymbol{c},\boldsymbol{b}))$ is a total function. Furthermore,

$$H^\infty(\boldsymbol{c},\boldsymbol{b}) \simeq 0_{\mathfrak{A}} \quad \Longrightarrow \quad |\boldsymbol{c}|_I < \sigma \implies \boldsymbol{c} \in \mathsf{Dom}\ I^{<\sigma} \subseteq \mathsf{Dom}\ I^\infty,$$

so $I^\infty \upharpoonright Z_{\lambda\, \boldsymbol{c}\,.\, H^\infty(\boldsymbol{c},\boldsymbol{b})} \subseteq I^{<\sigma}$. The hypotheses of (2) mean that $I(\boldsymbol{a}, I^{<\sigma}) \simeq I^\sigma(\boldsymbol{a}) \uparrow$ and we have all of the conditions satisfied to guarantee

$$G^\infty(\boldsymbol{a},\boldsymbol{b}) \simeq G(\boldsymbol{a},\boldsymbol{b},G^\infty,H^\infty) \simeq \Delta_I(\boldsymbol{a}, I^\infty, \lambda\, \boldsymbol{c}\,.\, H^\infty(\boldsymbol{c},\boldsymbol{b})) \simeq 0_{\mathfrak{A}},$$

as desired. The proof of (4) is parallel.

Now we have

$$C_{I,J}(\boldsymbol{a},\boldsymbol{b}) \simeq \begin{cases} 1_{\mathfrak{A}}(I^\infty(\boldsymbol{a})), & \text{if } G^\infty(\boldsymbol{a},\boldsymbol{b}) \simeq 1_{\mathfrak{A}}; \\ 0_{\mathfrak{A}}(J^\infty(\boldsymbol{a})), & \text{if } G^\infty(\boldsymbol{a},\boldsymbol{b}) \simeq 0_{\mathfrak{A}}, \end{cases}$$

where $1_{\mathfrak{A}}$ and $0_{\mathfrak{A}}$ denote constant unary functions. □

7.13 Corollary. *If $\mathfrak{A}$ is normal, then*

(i) $\mathsf{Env}_{\mathsf{Rec}}(\mathfrak{A})$ *is closed under finite union;*

(ii) $R \in \mathsf{Sec}_{\mathsf{Rec}}(\mathfrak{A})$ *iff both R and $A^k \sim R$ belong to* $\mathsf{Env}_{\mathsf{Rec}}(\mathfrak{A})$.

Proof. For (i), let $R = \mathsf{Dom}\ \mathsf{Rec}(\mathcal{I})$ and $S = \mathsf{Dom}\ \mathsf{Rec}(\mathcal{J})$ be two $\mathfrak{A}$-semi-recursive relations on A. Then easily $R \cup S = \mathsf{Dom}\ \lambda\, \boldsymbol{a}\,.\, C_{\mathcal{I},\mathcal{J}}(\boldsymbol{a},\boldsymbol{a})$ and is thus $\mathfrak{A}$-semi-recursive. For (ii), suppose that $R = \mathsf{Dom}\ \mathsf{Rec}(\mathcal{I})$ and $A^k \sim R = \mathsf{Dom}\ \mathsf{Rec}(\mathcal{J})$. Then easily, $C_{\mathcal{I},\mathcal{J}}(\boldsymbol{a},\boldsymbol{a})$ is the characteristic function of R, which is thus $\mathfrak{A}$-recursive. □

For first-order structures with arithmetic (for which we have universal functions), this result may also be proved by establishing a number selection theorem. We first introduce a modified indexing of the partial $\mathfrak{A}$-recursive functions:

$$\{\!\{\langle\, c, s\,\rangle\}\!\}^{\mathfrak{A}}_{\mathsf{Rec}}(x) \simeq \{c\}^{\mathfrak{A}}_{\mathsf{Rec}}(s,x),$$

for which we can prove the Second Recursion Theorem. We follow here the presentation of [Ke-Mo].

7.14 Proposition. *For any first-order structure $\mathfrak{A}$ with arithmetic, there exist primitive recursive functions $S_n^{k,l}$ such that for all $b, m_0, \ldots, m_{n-1} \in \omega$,*

$$\{\{S_n^{k,l}(b, \boldsymbol{m})\}\}^{\mathfrak{A}}_{\mathsf{Rec}}(x) \simeq \{\{b\}\}^{\mathfrak{A}}_{\mathsf{Rec}}(\boldsymbol{m}, x).$$

Proof. There exists an index $\bar{d}$ such that

$$\{\bar{d}\}^{\mathfrak{A}}_{\mathsf{Rec}}(\langle\, b, \boldsymbol{m}\,\rangle, x) \simeq \{\{b\}\}^{\mathfrak{A}}_{\mathsf{Rec}}(\boldsymbol{m}, x).$$

Now if $S_n^{k,l}(b, \boldsymbol{m}) := \langle\, \bar{d}, \langle\, b, \boldsymbol{m}\,\rangle\,\rangle$, we have

$$\{\{S_n^{k,l}(b, \boldsymbol{m})\}\}^{\mathfrak{A}}_{\mathsf{Rec}}(x) \simeq \{\bar{d}\}^{\mathfrak{A}}_{\mathsf{Rec}}(\langle\, b, \boldsymbol{m}\,\rangle, x) \simeq \{\{b\}\}^{\mathfrak{A}}_{\mathsf{Rec}}(\boldsymbol{m}, x),$$

as desired. □

7.15 Second Recursion Theorem. *For any first-order structure $\mathfrak{A}$ with arithmetic, for any partial $\mathfrak{A}$-recursive functional F, there exists an index $\bar{b}$ such that for all x,*

$$\{\{\bar{b}\}\}^{\mathfrak{A}}_{\mathsf{Rec}}(x) \simeq F(\bar{b}, x).$$

Proof. As usual, set $\bar{b} = S_1^{k,l}(\bar{c}, \bar{c})$, where $\{\{\bar{c}\}\}^{\mathfrak{A}}_{\mathsf{Rec}}(b, x) \simeq F(S_1^{k,l}(b, b), x)$. □

7.16 Number Selection Theorem. *For any first-order structure with arithmetic, for each partial $\mathfrak{A}$-recursive functional F there exists a partial $\mathfrak{A}$-recursive functional Sel_F such that*

$$(\exists n \in \omega) F(n, \boldsymbol{a}) \downarrow \quad \Longrightarrow \quad \mathsf{Sel}_F(\boldsymbol{a}) \downarrow \quad \textit{and} \quad F(\mathsf{Sel}_F(\boldsymbol{a}), a) \downarrow .$$

Proof. Since $\mathfrak{A}$ is first-order, it is normal and the stage-comparison functional is partial $\mathfrak{A}$-recursive. Let $\mathcal{I}$ be an iterable sequence of $\mathsf{Expl}_{\mathfrak{A}}$ functionals such that $I_0^{\infty}(b, a, x) \simeq \{\{b\}\}^{\mathfrak{A}}_{\mathsf{Rec}}(a, x)$ is the appropriate universal functional and fix an index $\bar{b}$ for F. Set

$$G(c, a, x) \simeq \begin{cases} 0, & \text{if } a \in \omega \text{ and } \mathcal{C}_{\mathcal{I},\mathcal{I}}(\bar{b}, a, x, c, a+1, x) \simeq 0; \\ \{\{c\}\}^{\mathfrak{A}}_{\mathsf{Rec}}(a+1, x) + 1, & \text{if } a \in \omega \text{ and } \mathcal{C}_{\mathcal{I},\mathcal{I}}(\bar{b}, a, x, c, a+1, x) \simeq 1; \\ 0, & \text{if } a \notin \omega. \end{cases}$$

By the Second Recursion Theorem, choose $\bar{c}$ such that

$$I_0^{\infty}(\bar{c}, a, x) \simeq \{\{\bar{c}\}\}^{\mathfrak{A}}_{\mathsf{Rec}}(a, x) \simeq G(\bar{c}, a, x).$$

Suppose that $F(n, x) \downarrow$ so $I_0^{\infty}(\bar{b}, n, x) \downarrow$ and hence $\mathcal{C}_{\mathcal{I},\mathcal{I}}(\bar{b}, n, x, c, n+1, x) \simeq 0$ or 1. In the first case $G(\bar{c}, n, x) \simeq 0$ and in the second, since $|\bar{c}, n+1, x|_{\mathcal{I}} < |\bar{b}, n, x|_{\mathcal{I}}$, $G(\bar{c}, n+1, x) \simeq \{\{\bar{c}\}\}^{\mathfrak{A}}_{\mathsf{Rec}}(n+1, x) \downarrow$. Furthermore, it follows easily that if $G(\bar{c}, m, x) \downarrow$, then also $G(\bar{c}, m', x) \downarrow$ for all $m' < m$. Hence in any case $G(\bar{c}, 0, x) \downarrow$, say $G(\bar{c}, 0, x) \simeq p$. Then clearly $G(\bar{c}, 1, x) \simeq p - 1, \ldots, G(\bar{c}, p, x) \simeq 0$, and this must be due to the first clause — that is $\mathcal{C}_{\mathcal{I},\mathcal{I}}(\bar{b}, p, x, \bar{x}, p+1, x) \simeq 0$. Hence $F(p, x) \simeq I_0^{\infty}(\bar{b}, p, x) \downarrow$ and thus $\mathsf{Sel}_F(x) :\simeq G(\bar{c}, 0, x)$ has the desired property. □

As a final example we show

7.17 Theorem. *For any first-order structure,* $\mathfrak{A}^{+} = (\mathfrak{A}, =_A, \mathsf{E}^\sharp_A)$ *is normal.*

Proof. Let G^0 and G^1 be the functionals defined by

$$G^i(a, f, \delta) \simeq \begin{cases} i_{\mathfrak{A}}, & \text{if } \delta(a) \downarrow \neq 0_{\mathfrak{A}}; \\ \mathsf{IsOne}_{\mathfrak{A}}(f(a)), & \text{if } \delta(a) \simeq 0_{\mathfrak{A}}. \end{cases}$$

Then we shall show that

$$\Delta_{\mathsf{E}^\sharp_A}(f, \delta) \simeq \begin{cases} 1_{\mathfrak{A}}, & \text{if } \mathsf{E}^\sharp_A(G^0_{f,\delta}) \simeq 1_{\mathfrak{A}}; \\ \mathsf{IsZero}_{\mathfrak{A}}(\mathsf{E}^\sharp_A(G^1_{f,\delta}), & \text{if } \mathsf{E}^\sharp_A(G^0_{f,\delta}) \simeq 0_{\mathfrak{A}}; \end{cases}$$

is a normalizing function for $\mathsf{E}^\sharp_A$ which is $\mathfrak{A}^{+}$-explicit. (Recall that $G^i_{f,\delta} = \lambda a \, . \, G^i(a, f, \delta)$).

Suppose first that $\mathsf{E}^\sharp_A(f \upharpoonright Z_\delta) \simeq 1_{\mathfrak{A}}$. Then

$$\exists a \, [f(a) \simeq 1_{\mathfrak{A}} \wedge \delta(a) \simeq 0_{\mathfrak{A}}] \qquad \text{so} \qquad \exists a \, [G^0(a, f, \delta) \simeq 1_{\mathfrak{A}}],$$

whence $\mathsf{E}^\sharp_A(G^0_{f,\delta}) \simeq 1_{\mathfrak{A}}$ and thus $\Delta_{\mathsf{E}^\sharp_A}(f, \delta) \simeq 1_{\mathfrak{A}}$.

Next, if $\mathsf{E}^\sharp_A(f \upharpoonright Z_\delta) \simeq 0_{\mathfrak{A}}$, then

$$\forall a \, [f(a) \simeq 0_{\mathfrak{A}} \wedge \delta(a) \simeq 0_{\mathfrak{A}}], \quad \text{so for i} = 0, 1, \quad \forall a \, [G^i(a, f, \delta) \simeq 0_{\mathfrak{A}}],$$

whence for $i = 0, 1$, $\mathsf{E}^\sharp_A(G^i_{f,\delta}) \simeq 0_{\mathfrak{A}}$ and thus $\Delta_{\mathsf{E}^\sharp_A}(f, \delta) \simeq 1_{\mathfrak{A}}$.

Now suppose that δ is total, $Z_\delta \subseteq \mathsf{Dom}\ F$ and $\mathsf{E}^\sharp_A(f \upharpoonright Z_\delta) \uparrow$. Since $\mathsf{E}^\sharp_A(f \upharpoonright Z_\delta) \not\simeq 1_{\mathfrak{A}}$, we have

$$\forall a \, [\delta(a) \simeq 0_{\mathfrak{A}} \implies f(a) \not\simeq 1_{\mathfrak{A}}], \quad \text{so since } \delta \text{ is total}, \quad \forall a \, [G^0(a, f, \delta) \simeq 0_{\mathfrak{A}}],$$

whence $\mathsf{E}^\sharp_A(G^0_{f,\delta}) \simeq 0_{\mathfrak{A}}$. Since $\mathsf{E}^\sharp_A(f \upharpoonright Z_\delta) \not\simeq 0_{\mathfrak{A}}$,

$$\neg \forall a \, [\delta(a) \simeq 0_{\mathfrak{A}} \wedge f(a) \downarrow \neq 1_{\mathfrak{A}}] \quad \text{so} \quad \text{either } \exists a \, [\delta(a) \downarrow \neq 0_{\mathfrak{A}}] \text{ or } \exists a \, [\delta(a) \simeq 0_{\mathfrak{A}} \wedge f(a) \simeq 1_{\mathfrak{A}}].$$

In either case $\exists a \, [G^1(a, f, \delta) \simeq 1_{\mathfrak{A}}]$, whence $\mathsf{E}^\sharp_A(G^1_{f,\delta}) \simeq 1_{\mathfrak{A}}$, and thus $\Delta_{\mathsf{E}^\sharp_A}(f, \delta) \simeq 0_{\mathfrak{A}}$. $\square$

8. Appendix

For completeness, we include here sketches of proofs of two technical results used in the proof of Theorem 4.5.

8.1 Proposition. *The class of simply partial $\mathfrak{A}$-recursive functionals is closed under substitution.*

Proof. To keep the notation under control, we shall prove explicitly a special case and hope that the extension to the general case is clear. Let G and H be simply partial $\mathfrak{A}$-recursive, say

$$G = I^\infty = \mathsf{Rec}(I, J) \qquad \text{and} \qquad H = K^\infty = \mathsf{Rec}(K),$$

with I, J, and K, all $\mathfrak{A}$-explicit, and

$$F(a, x) \simeq G(a, x, H_x) \simeq G(a, x, \lambda\, c\,.\, H(c, x)).$$

Let $({}^*I, {}^*J, {}^*K)$ be the following iterable sequence of $\mathfrak{A}$-explicit functionals:

$$\begin{aligned} {}^*I(a, x, i, j, k) &\simeq I(a, x, k, i, j) \\ {}^*J(b, x, i, j, k) &\simeq J(b, x, k, i, j) \\ {}^*K(c, x, i, j, k) &\simeq K(c, x, k) \end{aligned}$$

.

We claim that $F = {}^*I^\infty = \mathsf{Rec}({}^*I, {}^*J, {}^*K)$ and is thus partial $\mathfrak{A}$-recursive — in other words, that for all x,

$$(1) \qquad {}^*I^\infty_x = I^\infty_{x, H_x}.$$

It will suffice to show that for all σ,

$$(2) \qquad {}^*K^\sigma = K^\sigma \qquad \text{whence} \quad {}^*K^\sigma \subseteq H$$

and for all σ and x,

$$(3) \qquad {}^*I^\sigma_x \subseteq I^\sigma_{x, H_x} \subseteq {}^*I^\infty_x \qquad \text{and} \qquad {}^*J^\sigma_x \subseteq J^\sigma_{x, H_x} \subseteq {}^*J^\infty_x.$$

The intuition behind (3) (for I) is that ${}^*I^\sigma(a, x)$ is obtained by applying $I(a, x, {}^*K^{<\sigma}, \ldots)$ to earlier stages. By (2) this is 'weaker' than applying $I(a, x, H_x, \ldots)$, which is used to compute I^σ_{x, H_x}, but in the limit the result is the same.

The proof of (3) is also by induction; assume as induction hypothesis all of these inclusions for $\tau < \sigma$. Thus we have

$$(4) \qquad {}^*I^{<\sigma}_x \subseteq I^{<\sigma}_{x, H_x} \subseteq {}^*I^\infty_x \qquad \text{and} \qquad {}^*J^{<\sigma}_x \subseteq J^{<\sigma}_{x, H_x} \subseteq {}^*J^\infty_x.$$

Now if for some e,

$$^*I^\sigma(a,x) \simeq {}^*I(a,x,{}^*I_x^{<\sigma},{}^*J_x^{<\sigma},{}^*K_x^{<\sigma}) \simeq I(a,x,{}^*K_x^{<\sigma},{}^*I_x^{<\sigma},{}^*J_x^{<\sigma}) \simeq e,$$

then by monotonicity, (2), and the first and third inclusions of (4) we have

$$I^\sigma(a,x,H_x) \simeq I(a,x,H_x,I^{<\sigma}_{x,H_x},J^{<\sigma}_{x,H_x}) \simeq e,$$

which gives the first inclusion of (3). Now, by (2) and the second and fourth inclusions of (4), if $I^\sigma(a,x,H_x) \simeq e$, we have also

$$^*I^\infty(a,x) \simeq {}^*I(a,x,{}^*I_x^{\infty},{}^*J_x^{\infty},{}^*K_x^{\infty}) \simeq I(a,x,{}^*K_x^{\infty},{}^*I_x^{\infty},{}^*J_x^{\infty}) \simeq e,$$

and hence the second inclusion of (3). The corresponding calculations for J are similar. □

8.2 Proposition. *The class of simply partial $\mathfrak{A}$-recursive functionals is closed under recursion.*

Proof. We again prove a special case. Let I be simply partial $\mathfrak{A}$-recursive, say $I = J^\infty = \mathsf{Rec}(J,K)$, with J and K both $\mathfrak{A}$-explicit, and $F = I^\infty = \mathsf{Rec}(I)$. Let $({}^*I,{}^*J,{}^*K)$ be the following iterable sequence of $\mathfrak{A}$-explicit functionals:

$$\begin{aligned} {}^*I(a,x,i,j,k) &\simeq j(a) \\ {}^*J(a,x,i,j,k) &\simeq J(a,x,i,j,k) \\ {}^*K(b,x,i,j,k) &\simeq K(b,x,i,j,k). \end{aligned}$$

We claim that $F = {}^*I^\infty = \mathsf{Rec}({}^*I,{}^*J,{}^*K)$ and is thus simply partial $\mathfrak{A}$-recursive. To establish this it will suffice to show that for all σ,

$$^*I^\sigma \subseteq I^\sigma \subseteq {}^*I^\infty. \tag{1}$$

We need first to establish that for all σ and x,

$$J^\sigma_{x,{}^*I^\infty_x} \subseteq {}^*J^\infty_x \qquad \text{and} \qquad K^\sigma_{x,{}^*I^\infty_x} \subseteq {}^*K^\infty_x.$$

Assuming this for $\tau < \sigma$, if $J^\sigma(a,x,{}^*I^\infty_x) \simeq e$, then

$$^*J(a,x,{}^*I^\infty_x,J^{<\sigma}_{x,{}^*I^\infty_x},K^{<\sigma}_{x,{}^*I^\infty_x}) \simeq J(a,x,{}^*I^\infty_x,J^{<\sigma}_{x,{}^*I^\infty_x},K^{<\sigma}_{x,{}^*I^\infty_x}) \simeq J^\sigma(a,x,{}^*I^\infty_x) \simeq e,$$

so by the induction hypothesis and monotonicity, also

$$^*J^\infty(a,x) \simeq {}^*J(a,x,{}^*I^\infty_x,{}^*J^\infty_{x,{}^*I^\infty_x},{}^*K^\infty_{x,{}^*I^\infty_x}) \simeq e.$$

The calculation for K is analogous. Thus we have

$$J^\infty_{x,{}^*I^\infty_x} \subseteq {}^*J^\infty_x \qquad \text{and} \qquad K^\infty_{x,{}^*I^\infty_x} \subseteq {}^*K^\infty_x. \tag{2}$$

Now we prove (1) simultaneously by induction with

$$(3) \qquad {}^*J^\sigma_x \subseteq J^\sigma_{x,I^\sigma_x} \qquad \text{and} \qquad {}^*K^\sigma_x \subseteq K^\sigma_{x,I^\sigma_x}.$$

Assume as induction hypothesis that these hold for $\tau < \sigma$, so we have

$$(4) \qquad {}^*I^{<\sigma} \subseteq I^{<\sigma} \subseteq {}^*I^\infty,$$

and

$$(5) \qquad {}^*J^{<\sigma}_x \subseteq J^{<\sigma}_{x,I^{<\sigma}_x} \quad \text{and} \quad {}^*K^{<\sigma}_x \subseteq K^{<\sigma}_{x,I^{<\sigma}_x}.$$

Toward the first inclusion of (1), suppose that ${}^*I^\sigma(a,x) \simeq e$. Then

$${}^*J^{<\sigma}(a,x) \simeq {}^*I(a,x,{}^*I^{<\sigma}_x,{}^*J^{<\sigma}_x,{}^*K^{<\sigma}_x,) \simeq {}^*I^\sigma(a,x) \simeq e,$$

so by (5), also $J^{<\sigma}(a,x,I^{<\sigma}_x) \simeq e$, whence

$$I^\sigma(a,x) \simeq I(a,x,I^{<\sigma}_x) \simeq J^\infty(a,x,I^{<\sigma}_x) \simeq e,$$

as desired. Next, for the second inclusion of (1), suppose that

$$I^\sigma(a,x) \simeq e \quad \text{so also} \quad J^\infty(a,x,I^{<\sigma}_x) \simeq I(a,x,I^{<\sigma}_x) \simeq e.$$

Then by (4), $J^\infty(a,x,{}^*I^\infty_x) \simeq e$, so by (2), ${}^*J^\infty(a,x) \simeq e$. Hence

$${}^*I^\infty(a,x) \simeq {}^*I(a,x,{}^*I^\infty_x,{}^*J^\infty_x,{}^*K^\infty_x) \simeq {}^*J^\infty(a,x) \simeq e.$$

Finally, for (3), suppose that

$${}^*J^\sigma(a,x) \simeq {}^*J(a,x,{}^*I^{<\sigma}_x,{}^*J^{<\sigma}_x,{}^*K^{<\sigma}_x) \simeq e.$$

Then by (4) and (5), also

$$J^\sigma(a,x,I^{<\sigma}_x) \simeq J(a,x,I^{<\sigma}_x,J^{<\sigma}_{x,I^{<\sigma}_x},K^{<\sigma}_{x,I^{<\sigma}_x}) \simeq e,$$

whence also (by monotonicity), $J^\sigma(a,x,I^\sigma_x) \simeq e$ as desired. □

REFERENCES

[ASW] P. Aczel, H. Simmons, and S.S. Wainer (eds.), ***Proof Theory: Papers from the International Summer School, Leeds, 1990***, Cambridge University Press, Cambridge, x+306 pp., 1992

[Ba] J. Barwise (ed.), ***Handbook of Mathematical Logic***, North-Holland Pub Co., Amsterdam, New York, Oxford, xi+1165 pp., 1977.

[BORST] E. Boerger, W. Oberschelp, M.M. Richter, B. Schinzel, and W. Thomas (eds.), ***Computation and Proof Theory: Proceedings of the Logic Colloquium, vol. 2***, Springer-Verlag, Berlin, Heidelberg, New York (Lecture Notes in Mathematics 1104), viiii+475 pp., 1983

[BSS] L. Blum, M. Shub, and S. Smale, *On a Theory of Computation and Complexity over the Real Numbers: NP-Completeness, Recursive Functions, and Universal Machines*, ***Bull. Amer. Math. Soc.*** 21 (1989) 1-46.

[Cu] N.J. Cutland, ***Computability: An Introduction to Recursive Function Theory***, Cambridge Univ Press, Cambridge, x+251 pp., 1980.

[DW] F.R Drake and S.S. Wainer, ***Recursion Theory: its Generalizations and Applications***, London Mathematical Society Lecture Notes 45, Cambridge University Press, Cambridge, vi+319 pp., 1980

[En1] E. Engeler, ***Formal Languages: Automata and Structures***, Markham Publ Co., Chicago, vii+81 pp., 1968.

[En2] E. Engeler, *On the solvability of algorithmic problems*, in [RS] 231-251.

[Fe] J-E. Fenstad, ***General Recursion theory: An Axiomatic Approach***, Springer-Verlag, Berlin, Heidelberg, New York, xi+225 pp., 1980

[Fr] H.M. Friedman, *Algorithmic procedures, generalized Turing algorithms, and elementary recursion theory*, in [GY] 113-137.

[GY] R.O. Gandy and C.M.E. Yates (eds.), ***Logic Colloquium '69***, North-Holland, Amsterdam, xiv+457 pp., 1971

[HeIs] G.T. Herman and S.D. Isard, *Computability over arbitrary fields*, ***Jour. London Math. Soc.*** Ser. 2, 2 (1970) 73-79

[Hi] P.G. Hinman, ***Recursion-Theoretic Hierarchies***, Springer-Verlag, Berlin, Heidelberg, New York, xii+480 pp., 1978

[Ke-Mo] A.S. Kechris and Y.N. Moschovakis, *Recursion in Higher Types*, in [Ba] 681-737.

[Kl1] S.C. Kleene, *Recursive functionals and quantifiers of finite types I*, ***Trans. Amer. Math. Soc.*** 91 (1959) 1-52.

[Kl2] S.C. Kleene, *Recursive functionals and quantifiers of finite types II*, ***Trans. Amer. Math. Soc.*** 108 (1963) 106-142.

[Kl3] S.C. Kleene, *Turing-machine computable functionals of finite types I*, in [NST] 38-45.

[Kl4] S.C. Kleene, *Turing-machine computable functionals of finite types II*, ***Proc. London Math. Soc.*** (ser 3) 12 (1962) 245-258.

[Mi1] C. Michaux, *Une remarque à propos des machines sur* $\mathbf{R}$ *introduites par Blum, Shub, et Smale*, ***C.R. Acad. Sci. Paris, t.309, Série I*** (1989) 435-437.

[Mi2] C. Michaux, *Machines sur les réels et problèmes NP-complets*, ***Séminaire de Structures Algébriques Ordonnées, Prépubl. de l'equipe de logique math. de Paris 7*** 1990.

[Mo1] Y.N. Moschovakis, ***Elementary induction on abstract structures***, North-Holland, Amsterdam, x+218 pp., 1974

[Mo2] Y.N. Moschovakis, *Abstract recursion as a foundation for the theory of algorithms*, in [BORST] 289-364.

[Mo3] Y.N. Moschovakis, *The formal language of recursion*, ***J. Symb. Logic*** 54 (1989) 1216-1252.

[Mol] J. Moldestad, ***Computations in higher types***, Springer-Verlag, Berlin, Heidelberg, New York (Lecture Notes in Mathematics 574), iv+203 pp., 1977

[MS-HT1] J. Moldestad, V. Stoltenberg-Hansen, and J.V. Tucker, *Finite Algorithmic Procedures and Inductive Definability*, ***Math. Scand.*** 46 (1980) 62-76.

[MS-HT2] J. Moldestad, V. Stoltenberg-Hansen, and J.V. Tucker, *Finite Algorithmic Procedures and Computation Theories*, ***Math. Scand.*** 46 (1980) 77-94.

[MyO] J.P. Myers, Jr. and M.J. O'Donnell (eds.), ***Constructivity in Computer Science: Proceedings of the summer symposium, San Antonio, 1991***, Springer-Verlag, Berlin, Heidelberg, New York (Lecture Notes in Computer Science 613), x+247 pp., 1992

[NST] E. Nagel, P. Suppes, and A. Tarski (eds.), ***Proceedings of the 1st International Congress for Logic, Methodology, and the Philosophy of Science***, Stanford University Press, ix+661 pp., 1962

[Pl] R.A. Platek, ***Foundations of Recursion Theory***, PhD. dissertation, Stanford University, 1966.

[RS] H.E. Rose and J.C. Shepherdson, ***Logic Colloquium '73***, North-Holland, Amsterdam, viii+513 pp., 1975

[Sch] H. Schwichtenberg (ed.), ***Proof and Computation: Proceedings of the NATO Advanced Study Institute and Summer School, Marktoberdorf, 1993***, Springer-Verlag, Berlin, Heidelberg, New York, vi+470 pp., 1995

[Sh-St] J.C. Shepherdson and H.E. Sturgis, *Computability of recursive functions*, ***J. Assoc. for Comput. Mach.*** 10 (1963) 217-255.

[Tu] J.V. Tucker, *Computing in algebraic systems*, in [DW] 215-235.

[TuZu1] J.V. Tucker and J.I. Zucker, ***Program Correctness over Abstract Data Types***, North-Holland, Amsterdam, vii+212 pp., 1988

[TuZu2] J.V. Tucker and J.I. Zucker, *Examples of semicomputable sets of real and complex numbers*, in [MyO] 179-198.

[TuZu3] J.V. TUCKER AND J.I. ZUCKER, *Deterministic and nondeterministic computation and Horn programs on abstract data types*, ***Jour. of Logic Programming*** 13 (1992) 23-55.

[TuZu4] J.V. TUCKER AND J.I. ZUCKER, *Computable functions on stream algebras*, in [Sch] 397-437.

[TuZu5] J.V. TUCKER AND J.I. ZUCKER, *Provable computable selection functions on abstract structures*, in [ASW] 277-306.