Classifying Recursive Functions

Helmut Schwichtenberg

Draft of October 7, 1997 for the Handbook of Recursion Theory (ed. E. Griffor), North-Holland

1 Introduction

Ever since the recursive functions have been identified there was a challenge to measure their inherent computational complexity, or in Kleene's words [25] to "classify the recursive functions into a hierarchy, according to some general principle". The constructive ordinals of Church and Kleene lend themselves as an obvious scale for such classification attempts.

The most natural way to measure the 'complexity' of a recursive function by constructive ordinals is to consider recursion along recursive well-orderings of larger and larger order types. A positive result in this direction is due to Rósza Péter, who has shown that the class of functions definable with (substitution and) n-fold nested recursion grows with n increasing. One can view n-fold nested recursion as recursion on a special, 'natural' well-ordering of order type ω^n. In section 2 we will prove a theorem due to Myhill [39] and Routledge [49], which says that any recursive function can be defined using (Csillag-Kalmár) elementary operations and just one recursion on an elementary well-ordering of type ω. Myhill refers to this fact as "a stumblingblock in constructive mathematics"; in particular it implies that Péter's result essentially depends on the special form of the well-orderings used. With a similar method we prove that such a 'collapse' will also occur if we allow transfinite recursion for the definition of 0-1-valued functions only. Finally we prove a rather general result, which says that it is impossible to index the recursive functions in any 'reasonable' way by means of a Π^1_1-path through Kleene's system $\mathcal{O}$ of notations for constructive ordinals.

Therefore it seems necessary to look out for a notion of a 'standard' well-ordering of the natural numbers. But this turns out to be a very difficult task. Although an interesting attempt has been made by Zemke [71] and later been extended by Buchholz, Cichon and Weiermann [6], it seems fair to say that up to now no satisfactory such notion has been found. Therefore in this paper we restrict ourselves to 'canonical' well-orderings of concrete order types like ε_0. In this case the Cantor normal form can be used to define a canonical coding; moreover, one also has a canonical choice of fundamental sequences approximating the limit ordinals. Using these, a quite natural sequence F_α, $\alpha < \varepsilon_0$ of fast growing functions can be defined. Then one defines $\mathcal{E}_\alpha$ as the elementary closure of F_α; the $\mathcal{E}_\alpha$ form the *extended Grzegorczyk hierarchy*.

For the classes $\mathcal{E}_\alpha$ quite a number of interesting characterizations are known. First of all, $\mathcal{E}_\alpha$ is the class of functions computable by a register machine with time (i.e. number of computation steps) bounded by a finite iteration of the function F_α. Moreover, the functions in $\mathcal{E}_\alpha$ can be characterized by counting the number of recursions used and their order types, or else by transfinitely iterating the process of extending an effectively generated class of functions by enumeration (cf. [53]). So we have a quite satisfactory theory here.

However, by definition our smallest class $\mathcal{E}_0$ consists of the class of Csillag-Kalmár elementary functions and hence already contains exponentially growing functions. If one is interested in analyzing more 'feasible' notions of computation, then it is necessary to take into account the fact that numbers are represented as strings of numerals. This leads to a notion of 'bounded recursion on notation' introduced by Cobham [9]. We do not attempt to deal with this theory here, but refer the reader to the excellent survey of Clote [8] in the present Handbook.

So as a measure of complexity for recursive functions we use ordinals. We consider a recursive function to be given by a computation method or algorithm; hence closely connected to such a definition is a termination proof for the algorithm. Therefore it is to be expected that methods from proof theory are of central importance for the subject. More precisely, there is not much difference between classifying proofs of $\forall\exists$-theorems and definitions of recursive functions. For instance, by a classic result of Ackermann [1] and Kreisel [29] the ε_0-recursive functions are just those functions which are provably recursive in Peano arithmetic (cf. e.g. [15] or [57]), and the primitive recursive functions are just those definable by the subsystem of Peano arithmetic where induction is restricted to Σ_1-formulas; this result is due to Kreisel, Parsons, Mints [35] and Takeuti [67]. Since there is a recent survey on new developments along these lines in the paper by Fairtlough and Wainer [15], we do not go into this subject here.

One very interesting phenomenon needs to be mentioned: the role of ε_0 in classifying the provably recursive functions of Peano arithmetic is not as mandatory as one might think. It has been shown by Girard that the so-called slow growing hierarchy G_α, $\alpha <$ the Bachmann-Howard ordinal, yields the same growth as the fast growing hierarchy up to ε_0. This is interesting because a natural description of the Bachmann-Howard ordinal seems to require the least uncountable ordinal Ω. In [58] an attempt has been made to explain why uncountable ordinals play a role here.

The main subject of this article is the study of the interplay between recursion in higher types and transfinite recursion. To make the paper sufficiently self-contained we include a short introduction to the theory of partial continuous fnuctionals, based on Scott's notion of an information system. The partial continuous fnuctionals are central for any analysis of higher type computability which is based on the rather natural assumption that any computation ought to be finite. The reason is simply that they form the mathematically appropriate domain of a computable functional.

The total continuous functionals of Kleene-Kreisel [26, 30] can then be singled out from the partial continuous ones (cf. Ershov [13], Berger [4] and also Dag Normann's paper [43] in the present Handbook), and it seems best to define them that way.

Since any partial continuous functional Φ is the limit of finite approximations to it, it is straightforward to define computability: Φ is computable if and only if it is the limit of a computably enumerable set of finite approximations. This is an externally defined notion of computability, and the question arises whether there is an internal characterization. This is indeed the case: we will prove a result due to Plotkin [46] that a partial continuous functional is computable if and only if it can be defined explicitely from fixed point operators introduced into this framework by Platek [45], the parallel conditional pcond and a continuous approximation $\exists$ to the existential quantifier. We call such functionals recursive in pcond and $\exists$.

The term language with a fixed point operator leads us back to our original question of classifying recursive functions. To demonstrate the relevance of higher types for our subject, we first show that higher type iteration operators suffice to define all F_α, $\alpha < \varepsilon_0$, by application alone. We then show that it is possible to combine the power and elegance of the general fixed point operator with the desire to have better control over computational complexity by introduction of a bounded version of the fixed point operator. – Also for $\exists$ one can define a bounded version; together with (partial) primitive recursion this leads to subrecursive hierarchies over the partial continuous functionals, developed by Niggl [40, 42]. In [41] Niggl also studies another restricted notion of computability, generalizing Cook and Kapron's [10] typed while programs to partial continuous functionals.

We finally establish some kind of converse of the definability of the F_α, $\alpha < \varepsilon_0$, by means of higher type iteration operators. We show that finite types are not only sufficient to obtain the ε_0-recursive functions, but that their use can also be eliminated at the expense of higher ordinals. So we have a certain trade-off between the two concepts. The result itself is not new; however, here we sketch a different proof, by adapting a method of Buchholz.

The paper is organized as follows. Section 2 contains the collapsing results, and section 3 discusses the extended Grzegorczyk hierarchy. Section 4 gives an introduction to partial continuous functionals, and section 5 then contains some general material concerning computability in higher types, including definability of the F_α, $\alpha < \varepsilon_0$, by means of higher type iteration operators, and Plotkin's definability theorem. The discussion of bounded fixed point operators takes place in section 6, and the final section 7 concerns elimination of detours through higher types by transfinite recursion.

Acknowledgements

I would like to thank Peter Clote and Stan Wainer for their useful comments on various drafts of this paper.

2 Collapsing results

To set the stage, we prove the failure of some natural attempts to classify the recursive functions. Here we assume knowledge of certain introductory material from recursion theory, e.g. the Csillag-Kalmár definition of the elementary functions, Minsky's register machines and also Kleene's system $\mathcal{O}$ of ordinal notations, Π^1_1-sets and paths through $\mathcal{O}$.

2.1. *Register machines.* Minsky has introduced in [34] (see also Shepherdson and Sturgis [61]) a type of idealized computing machines now called register machines. These machines allow a rather direct and perspicuous proof that all recursive functions are computable. In particular one can prove in the well-known way the following theorem.

Theorem. *(Normal form theorem). Let f be a unary recursive function, p be the Gödel number of a register machine computing f and $s_f(x)$ the number of steps performed by this machine when computing $f(x)$. Then f can be written in the form*

$$f(x) = D(C(p, x, s_f(x)))$$

with elementary functions D ('decoding function') and C ('configuration function').
□

2.2. Now we can prove our first collapse result, due to Myhill [39] and Routledge [49].

Theorem. *For any recursive function f we can find an elementary well-ordering $\prec$ of the natural numbers with order type ω and a recursive function h such that f is elementary in h, and h can be defined in the form*

$$\begin{aligned} &h(0) = 0, \\ &h(u) = 1 + h(g(u)) \quad \textit{for } u \neq 0, \end{aligned}$$

with g an elementary function such that $g(u) \prec u$ for $u \neq 0$.

Proof. It clearly suffices to prove the theorem for unary functions f. Let p be the Gödel number of a register machine computing f and let $s_f(x)$ be the number of steps performed by this machine when computing $f(x)$; we may assume $s_f(x) \geq 1$ for all x. Note first that the relation $t < s_f(x)$ is elementary, since

$$t < s_f(x) \leftrightarrow C(p, x, t) \neq C(p, x, t+1)$$

with C the configuration function from 2.1. The pairs (t, x) with $t < s_f(x)$ can be well-ordered by

$$(t_1, x_1) \ll (t_2, x_2) \leftrightarrow x_1 < x_2 \vee (x_1 = x_2 \wedge t_1 > t_2).$$

Let B be the image of all these pairs under the mapping $\lambda t\lambda x(\pi(t,x)+1)$, i.e.

$$B(u) :\leftrightarrow \exists t{\le}u\exists x{\le}u(t < s_f(x) \wedge \pi(t,x)+1 = u)$$

Here π is an elementary pairing function with elementary inverses π_1, π_2. Clearly B is elementary. The 'induced' well-ordering on B can be extended easily to an elementary well-ordering $\prec$ on all natural numbers with least element 0 (note that $\neg B(0)$, $B(1)$). Let

$$\begin{aligned}
b(u) &:= \mu v{\le}u[B(v) \wedge \forall w{\le}u(B(w) \rightarrow w \le v)],\\
t(u) &:= \pi_1(b(u) \dot{-} 1),\\
x(u) &:= \pi_2(b(u) \dot{-} 1).
\end{aligned}$$

Then we define $\prec$ by

$$\begin{aligned}
u \prec v \leftrightarrow [u = 0 \wedge v \ne 0] \vee [u \ne 0 \wedge (&x(u) < x(v) \vee\\
&(x(u) = x(v) \wedge t(u) > t(v)) \vee\\
&(x(u) = x(v) \wedge t(u) = t(v) \wedge u < v))].
\end{aligned}$$

Now consider the function

$$g(u) := \begin{cases} \pi(t(u)+1, x(u))+1 & \text{if } B(u) \text{ and } t(u)+1 < s_f(x(u))\\ 0 & \text{otherwise.}\end{cases}$$

Then clearly we have $g(u) \prec u$ for all $u \ne 0$, and from $\pi(0,x)+1$ we come to 0 by exactly $s_f(x)$ applications of g. Moreover g is elementary. So if we define h by

$$\begin{aligned}
&h(0) = 0,\\
&h(u) = 1 + h(g(u)) \quad \text{for } u \ne 0,
\end{aligned}$$

then we have

$$s_f(x) = h(\pi(0,x)+1).$$

Now the claim follows, since by the normal form theorem 2.1 we have

$$f(x) = D(C(p, x, s_f(x))). \qquad \square$$

2.3. It is tempting to try to avoid this 'collapse' by allowing bounded recursion only. This may seem particularly promising, since Rósza Péter has proved that bounded multiple recursion does not lead out of the primitive recursive functions [44, p. 94]. However, the collapse cannot be avoided in this simple way.

THEOREM. *For any recursive* 0-1*-valued function f we can find an elementary well-ordering $\prec$ of the natural numbers with order type ω and a recursive* 0-1*-function h such that f is elementary in h, and h can be defined in the form*

$$\begin{aligned}
&h(0) = 0,\\
&h(u) = g_0(u, h(g(u))) \quad \textit{for } u \ne 0,\\
&h(u) \le 1,
\end{aligned}$$

with g, g_0 elementary functions such that $g(u) \prec u$ for $u \neq 0$.

PROOF. We proceed as in the previous proof, up to and including the definition of g. Then let

$$h(0) := 0,$$
$$h(u) := \begin{cases} h(g(u)) & \text{if } g(u) \neq 0 \\ \mathrm{sg}(D(C(p, x(u), t(u) + 1))) & \text{otherwise.} \end{cases}$$

Then $h(u) \leq 1$ and we have $f(x) = h(\pi(0, x) + 1)$ and hence the claim. □

2.4. *A general collapse result.* We now prove a rather general result, which says that it is impossible to index the recursive functions in any 'reasonable' way by means of a Π^1_1-path through Kleene's system $\mathcal{O}$ of notations for constructive ordinals. – I have learned this result from Stan Wainer, who in turn attributed it to some unpublished work of Yiannis Moschovakis.

We will need in this section some knowledge of Kleene's system $\mathcal{O}$, Π^1_1-sets and the like (cf. Shoenfield [62], Rogers [24], Sacks [50], Hinman [23] or other papers in the present Handbook). For definiteness, here is a list of what we need. A subset P of $\mathcal{O}$ is called a *path* through $\mathcal{O}$ if

(P1) P is linearly ordered by $<_{\mathcal{O}}$,

(P2) $b \in P$ and $a <_{\mathcal{O}} b$ implies $a \in P$, and

(P3) any constructive ordinal is denoted by a $b \in P$.

We will make use of the following facts.

(F1) There is a recursively enumerable relation $<'_{\mathcal{O}}$ extending $<_{\mathcal{O}}$, such that for any $b \in \mathcal{O}$ we have
$$a <_{\mathcal{O}} b \iff a <'_{\mathcal{O}} b.$$

(F2) (Feferman, Spector). There is a Π^1_1-path through $\mathcal{O}$.

(F3) There is no Σ^1_1-path through $\mathcal{O}$.

Let us now formulate what we mean by a 'reasonable' hierarchy. First, it should be indexed by a path P through $\mathcal{O}$, i.e. be of the form $(f_a)_{a\in P}$. Moreover, the property $R(a, x, y)$ that the a-th function f_a at argument x has value y should not be too complex. By this we mean that at least it should be inductively definable from arithmetical (or even Π^1_1-) relations, hence R itself should be a Π^1_1- relation. So let us assume that we have a Π^1_1- relation R satisfying

$$\forall a{\in}P \forall x \exists! y\, R(a, x, y)$$

and define $f_a(x)$ to be the unique y such that $R(a, x, y)$. We then require the following properties of our hierarchy $(f_a)_{a\in P}$.

(H1) $\forall a, b \in P(a \neq b \to f_a \neq f_b)$.

(H2) $\forall a \in P(f_a \in \text{REC})$.

(H3) $\forall f \in \text{REC} \exists a \in P(f = f_a)$.

Here REC denotes the set of all unary total recursive functions. So (H1)-(H3) expresses that our assumed hierarchy $(f_a)_{a\in P}$ provides a unique indexing of all $f \in \text{REC}$ by means of the path P through $\mathcal{O}$.

We now show that from all these assumptions we can derive a Σ^1_1-definition of P, in the form

$$a \in P \leftrightarrow \exists\{e\} \in \text{REC} \forall b \in P[\forall x R(b, x, \{e\}(x)) \to a <'_{\mathcal{O}} b]. \qquad (*)$$

Clearly the right hand side has Σ^1_1-form, since $\{e\} \in \text{REC}$ is arithmetical, the Π^1_1-relations P and R appear as premises and $<'_{\mathcal{O}}$ in the conclusion is recursively enumerable by (F1). Since by (F3) there is no Σ^1_1-definition of P, we have the desired contradiction.

It remains to prove $(*)$. $\to$. Let $a \in P$. Then by (P3) there is a $b \in P$ such that $a <_{\mathcal{O}} b$, hence $f_b \in \text{REC}$ by (H2). Pick e such that $f_b = \{e\}$. Then by (H1) b is uniquely determined by $f_b = \{e\}$, i.e. by the Π^1_1-relation $\forall x R(b, x, \{e\}(x))$. Since $b \in P \subseteq \mathcal{O}$, we also have $a <'_{\mathcal{O}} b$ by (F1).

$\leftarrow$. Let $\{e\} \in \text{REC}$ such that $\forall b \in P[f_b = \{e\} \to a <'_{\mathcal{O}} b]$. By (H3) there is a $b \in P$ such that $f_b = \{e\}$, and again by (H1) b is uniquely determined. Since $b \in P \subseteq \mathcal{O}$, from $a <'_{\mathcal{O}} b$ we can conclude $a <_{\mathcal{O}} b$ by (F1), and hence $a \in P$ by (P2).

3 The extended Grzegorczyk hierarchy

From now on we restrict attention to 'canonical' well-orderings of concrete order types like ε_0, Γ_0 or the Bachmann-Howard ordinal. We assume some knowledge of such ordinals, in particular of their Cantor normal form.

3.1. The *fast growing* functions F_α, $\alpha < \varepsilon_0$, are defined as follows.

$$\begin{aligned} F_0(x) &:= 2^x, \\ F_{\alpha+1}(x) &:= F_\alpha^{(x)}(x) \quad (F_\alpha^{(x)} \text{ } x\text{-th iterate of } F_\alpha), \\ F_\lambda(x) &:= F_{\lambda[x]}(x). \end{aligned}$$

Here the fundamental sequence $\lambda[x]$ for limit numbers $\lambda < \varepsilon_0$ and $x \in \mathbb{N}$ is defined in a quite natural way. First note that any such limit number can be written uniquely in the form $\lambda = \omega^{\alpha_n} + \ldots + \omega^{\alpha_0}$ with $\lambda > \alpha_n \geq \ldots \geq \alpha_0 > 0$. Then we let

$$\lambda[x] := \begin{cases} \omega^{\alpha_n} + \ldots + \omega^{\alpha_1} + \omega^{\alpha_0 - 1} \cdot x & \text{if } \alpha_0 \text{ is a successor} \\ \omega^{\alpha_n} + \ldots + \omega^{\alpha_1} + \omega^{\alpha_0[x]} & \text{if } \alpha_0 \text{ is a limit.} \end{cases}$$

There are many variants of the fast growing functions, which have essentially the same rate of growth. We only mention the *Hardy*-functions [22], which are defined by

$$\begin{aligned} H_0(x) &:= x, \\ H_{\alpha+1}(x) &:= H_\alpha(x+1), \\ H_\lambda(x) &:= H_{\lambda[x]}(x). \end{aligned}$$

A detailed comparison of closely related functions can be found in Fairtlough and Wainer [15].

3.2. We now define the *extended Grzegorczyk hierarchy*. Let $\mathcal{E}_\alpha$ be the elementary closure of F_α, i.e. the least class of functions containing the F_α and some initial functions ($U^i_n = \lambda x_1, \ldots, x_n.x_i$ (for $1 \leq i \leq n$), $C^i_n = \lambda x_1, \ldots, x_n.i$ (for $n \geq 0$, $i \geq 0$), $\lambda x, y.x+y$ and $\lambda x, y.x \dot{-} y$), that is closed against (simultaneous) substitution and bounded sums and products. The ε_0-*recursive functions* are defined to be the functions in

$$\bigcup_{\alpha<\varepsilon_0} \mathcal{E}_\alpha.$$

One of the classic results of the subject is the identification due to Ackermann [1] and Kreisel [29] of the ε_0-recursive functions as the provably recursive functions of arithmetic. In fact, there is quite an intimate relationship between proof theory and subrecursive hierarchies. However, since there is a recent survey on new developments along these lines in the paper by Fairtlough and Wainer [15], we do not go into this subject here. Another quite interesting approach can be found in Friedman and Sheard [17].

Also for the particular classes $\mathcal{E}_\alpha$ quite a number of interesting characterizations are known. First of all, $\mathcal{E}_\alpha$ is the class of functions computable by a register machine with time (i.e. number of computation steps) bounded by a finite iteration of the function F_α. Moreover, the functions in $\mathcal{E}_\alpha$ can be characterized by counting the number of recursions used and their order types, or else by transfinitely iterating the process of extending an effectively generated class of functions by enumeration (cf. [53]). Also proof theoretic characterizations of these classes are known (e.g. [54] and Takeuti [68]). So we have a quite satisfactory theory here.

3.3. One might ask to what extent ε_0 is typical for the set of recursive functions considered, i.e. the ε_0-recursive functions and hence the functions provably recursive in arithmetic. It seems that the results mentioned provide strong evidence that this is the case. However, this impression is not correct: Girard has shown in [19] that one might as well associate a far bigger ordinal with the functions provably recursive in arithmetic, the Bachmann-Howard ordinal. To this end, Girard has introduced the so-called slow growing hierarchy G_α, $\alpha <$ the Bachmann-Howard ordinal:

$$\begin{aligned} G_0(x) &:= x, \\ G_{\alpha+1}(x) &:= G_\alpha(x)+1, \\ G_\lambda(x) &:= G_{\lambda[x]}(x). \end{aligned}$$

This hierarchy catches up with the F_α only at the Bachmann-Howard ordinal. However, the natural description of this ordinal seems to need the least uncountable ordinal Ω, and one might wonder why Ω can play a role in a characterization of the provably recursive functions of arithmetic. In [58] an attempt has been made to explain why this is so.

3.4. The subrecursive classifications in terms of the fast growing hierarchy are due to Grzegorczyk [21] for $\alpha < \omega$, to Robbin [47] and the author [52] for $\alpha < \omega^\omega$ and generally to Löb and Wainer [32, 33], Wainer [69] and the author [53]. From the many other more recent contributions to the subject we only mention Rose's book [48] and the papers by Buchholz and Wainer [7], Fairtlough and Wainer [15], Friedman and Sheard [17] and Weiermann [70].

4 Partial continuous functionals

In the rest of this paper we will consider what happens when higher types are taken into account. This means that as arguments not only numbers are allowed, but also functions and even functionals of any finite type, where for now we take types to be built from the type ι of natural numbers by means of an operation $\rho \to \sigma$ to form function types. So for any type ρ a set D_ρ of objects of type ρ must be given. The precise definition of D_ρ depends on the particular approach to higher type computability, but in most cases we have $D_0 = \mathbb{N}$ and $D_{\rho\to\sigma}$ is a set of (possibly partial) functions from D_ρ into D_σ. In the present section we give a short exposition of the theory of partial continuous functionals, in order to make the paper sufficiently self-contained.

NOTATION. We will normally use $f, g, F, G, \dots$ for total and $\varphi, \psi, \Phi, \Psi, \dots$ for partial functions and functionals.

4.1. *Historical comments.* Let us first review Kleene's notion of partial recursive functionals, introduced in [27]. In his approach, D_ρ is the set HT_ρ of all 'hereditarily total' functionals of type ρ:

$$\begin{aligned} \mathrm{HT}_0 &:= \mathbb{N}, \\ \mathrm{HT}_{\rho\to\sigma} &:= \{\, f\colon \mathrm{HT}_\rho \to \mathrm{HT}_\sigma \mid f \text{ total} \,\}. \end{aligned}$$

Let $\mathrm{HT} := \bigcup_\rho \mathrm{HT}_\rho$. Kleene only considered the special cases $\sigma_0 := \iota$ and $\sigma_{n+1} := \sigma_n \to \iota$, but this is not an essential restriction; Gandy and Hyland [18] have developed Kleene's approach for the more general types considered here.

Kleene gave an inductive definition of the *partial recursive functionals* on these domains, by means of certain schemata $S1, \dots, S9$ (cf. Normann's paper [43] in the present Handbook). He defined a *recursive functional* as a partial recursive functional which happens to be total. The class of partial recursive functionals has been studied in detail by Kleene and others. For type level 1 one obtains the ordinary partial recursive functions, and for type level 2 the functionals computable by Turing machines with input functions available as oracles.

If one uses the schemata $S1, \dots, S9$ to compute $\Phi(x_1, \dots, x_n)$, then one thinks of the higher type arguments as being given by oracles. This means that one is questioning the x_i's with certain computed functionals as arguments. Now the only scheme which properly uses an argument of a level ≥ 2 is $S8$. It says that for any recursive functional Ψ of an appropriate type also the following functional Φ is partial recursive.

$$\Phi(x_1, \dots, x_n) :\simeq x_1(\lambda y\, \Psi(y, x_1, \dots, x_n), x_2, \dots, x_n). \tag{$S8$}$$

Here we require that Φ is defined only for arguments $x_1, \dots, x_n$ such that the functional $\lambda y\, \Psi(y, x_1, \dots, x_n)$ is total; this must be assumed since x_1 is only defined for hereditarily total arguments. But this requirement has the undesired consequence that the partial recursive functionals are not closed under substitution. More precisely we have:

Lemma. *There are partial recursive functionals Ψ, F such that F is total and*

$$\varphi(x_1, \dots, x_n) := \Psi(\lambda y\, F(y, x_1, \dots, x_n), x_1, \dots, x_n)$$

is not partial recursive.

Proof. Let

$$\begin{aligned}
\chi(n,p) &:= \begin{cases} 0 & \text{if } \neg T(n,n,p) \\ \text{undefined} & \text{otherwise.} \end{cases} \\
\Psi(I,n) &:= I(\lambda p\, \chi(n,p)) \quad \text{by } (S8). \\
\varphi(n) &:= \Psi(0^2, n) \quad \text{with } 0^2(\alpha) := 0.
\end{aligned}$$

Then

$$\begin{aligned}
\varphi(n)\downarrow &\leftrightarrow \Psi(0^2, n)\downarrow \\
&\leftrightarrow 0^2(\lambda p\, \chi(n,p))\downarrow \\
&\leftrightarrow \forall p\, \chi(n,p)\downarrow \\
&\leftrightarrow \forall p\, \neg T(n,n,p).
\end{aligned}$$

Hence φ is not partial recursive. □

Restriction to recursive functionals leads to a stable class of functionals, which in particular is closed under substitution. For the recursive functionals many characterizations are known, including some using abstract machines (see e.g. Kleene [28]). However, one serious problem remains: Due to the scheme $S8$ computation trees are in general infinite. The resulting theory is therefore more a definability theory than a theory of computation, since in the latter the requirement that a computation is finite seems to be essential.

The pathology apparent in the lemma suggests that one should extend the domains HT_ρ in Kleene's theory, i.e. should also allow partial arguments. Platek

[45] (see also Moldestad [38]) has developed such a theory. However, the problem mentioned above about the infinite nature of the resulting computation trees remains.

In fact, this problem was identified quite early in the development of higher type recursion theory. Kreisel mentions it explicitly in [30, p. 104]. Consequently, he introduced the *continuous functionals*, and in the same volume Kleene [26] proposed a notion of *countable functionals*, using so-called *associates*. Both notions turned out to be essentially equivalent. However, it took considerably more time and effort to find the 'right' mathematical context for dealing with continuous functionals: this seems to be what is known today as Scott-Ershov domain theory [59, 60, 12, 14]. In this setting the appropriate *domains* of computable functionals have been identified as the partial continuous functionals. Moreover, the countable functionals of Kleene-Kreisel have been singled out as *total* objects by Ershov [13]. Note that they are defined now on *all* partial continuous functionals. An abstract, domain theoretic characterization of totality has been given by Berger in [3, 4]; this turned out to be quite useful for further generalizations, e.g. of the density theorem and the Kreisel-Lacombe-Shoenfield theorem. We refer to Stoltenberg-Hansen, Griffor and Lindström [63] as an excellent textbook on domain theory; in Chapter 8.3 it contains an exposition of Berger's approach.

4.2. To set the stage for our introduction to the theory of partial continuous functionals, we begin with a discussion of certain general principles which follow from an analysis of higher type computation.

We take it as a basic assumption that any computation has to be finite. Hence, if a partial function φ is an input for a computable functional Φ, then the computation can only make use of a finite subfunction φ_0 of φ. Similarly, if a functional Φ is used as an argument of a computable type three functional $\mathcal{H}$, then Φ can only be called upon finitely many times, and each time Φ's argument must be presented to Φ in an explicit form, e.g. as a finite set of argument-value pairs (n, m). Hence $\mathcal{H}$ can only make use of a 'finite approximation' of its argument Φ. Such a notion of a finite approximation can be defined in a similar fashion as we move up through the types, and using it we can formulate our first general principle as follows.

Principle of Finite Support. If $\mathcal{H}(\Phi)$ is defined with value n, then there is a finite approximation Φ_0 of Φ such that $\mathcal{H}(\Phi_0)$ is defined with value n.

If $\Phi(\varphi_0)$ is defined with value n and φ_0' extends the finite function φ_0, then clearly also $\Phi(\varphi_0')$ should be defined with the same value n. This notion of an extension can be carried up through the types: e.g., a finite approximation Φ_0' extends Φ_0 if for any pair (φ_0, n) in the latter there exists a pair (φ_0', n) in the former such that φ_0 extends φ_0' (note the reversed order). The notion of an extension also makes sense for arbitrary functionals, so we require

Monotonicity Principle. If $\mathcal{H}(\Phi)$ is defined with value n and Φ' extends Φ, then also $\mathcal{H}(\Phi')$ is defined with value n.

It is a consequence of these two principles that for a computable functional a finite approximation to its arguments suffices to find the value, and moreover that the functional itself is completely determined by its finite approximations. Hence it seems natural to require that

(i) the domain of a computable functional consists of all functionals that can be represented as limits of finite approximations (these are called partial continuous functionals), and

(ii) a functional is computable if and only if it is the limit of a computably enumerable set of finite approximations.

We postpone a discussion of the second requirement to the next section; in the present section we give an abstract, axiomatic formulation of the above principles, in terms of the so-called information systems of Scott [59, 60]. From these we will define the notion of a continuous functional of arbitrary finite type over $\mathbb{N}$.

4.3. *Information systems.* The basic idea of information systems is to provide an axiomatic setting to describe approximations of abstract objects (like functions or functionals) by concrete, finite ones. We do not attempt to analyze the notion of 'concreteness' or finiteness here, but rather take an arbitrary countable set A of 'bits of data' or 'tokens' as a basic notion to be explained axiomatically. In order to use such data to build approximations of abstract objects, we need a notion of 'consistency', which determines when the elements of a finite set of tokens are consistent with each other. We also need an 'entailment relation' between consistent sets X of data and single tokens a, which intuitively expresses the fact that the information contained in X is sufficient to compute the bit of information a. The axioms below are a minor modification of Scott's, due to Larsen and Winskel [31].

DEFINITION. An *information system* is a structure $(A, \mathrm{Con}, \vdash)$ where A is a countable set (the *tokens*), Con is a nonempty set of finite subsets of A (the *consistent* sets), and $\vdash$ is a subset of $\mathrm{Con} \times A$ (the *entailment relation*) which satisfy

1. If $X \subseteq Y \in \mathrm{Con}$ then $X \in \mathrm{Con}$;
2. If $a \in A$ then $\{a\} \in \mathrm{Con}$;
3. If $X \vdash a$ then $X \cup \{a\} \in \mathrm{Con}$;
4. If $X \in \mathrm{Con}$ and $a \in X$ then $X \vdash a$;
5. If $X, Y \in \mathrm{Con}$ and $\forall b{\in}Y(X \vdash b)$ and $Y \vdash c$, then $X \vdash c$.

We shall write $X \vdash Y$ to mean $\forall b{\in}Y(X \vdash b)$. Lower case x, y, z will denote subsets of A, and upper case X, Y, Z will normally be used for finite subsets of A.

DEFINITION. The *objects* or *ideals* of an information system $\mathbf{A} = (A, \textsc{Con}, \vdash)$ are defined to be those subsets z of A which satisfy

1. $X \subseteq^{\text{fin}} z$ implies $X \in \textsc{Con}$ (z is 'consistent');

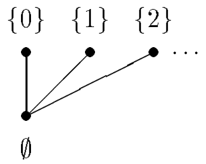
2. $X \subseteq^{\text{fin}} z$ and $X \vdash a$ implies $a \in z$ (z is 'deductively closed').

The set of all objects of $\mathbf{A}$ is written $|\mathbf{A}|$.

EXAMPLES. Any countable set A can be turned into a *flat* information system $\mathbf{A}$ by letting the set of tokens be A, and

$$\textsc{Con} := \{\emptyset\} \cup \{\, \{a\} \mid a \in A \,\} \qquad \text{and} \qquad X \vdash a \ :\equiv\ a \in X.$$

For $A = \mathbb{N}$ we have the following picture of the $\textsc{Con}$-sets.



In this case the objects are just the elements of $\textsc{Con}$. Note that if instead we took $\textsc{Con}$ to be the set of all finite subsets of A, then the objects would be all subsets of A.

Another rather important example is the following, which concerns approximations of functions from a countable set A into a countable set B. The tokens are the pairs (a, b) with $a \in A$ and $b \in B$, and

$$\begin{aligned}
&\textsc{Con} := \{\, \{(a_0, b_0), \dots, (a_{k-1}, b_{k-1})\} \mid \forall i, j{<}k\, (a_i = a_j \to b_i = b_j) \,\},\\
&X \vdash (a, b) \ :\equiv\ (a, b) \in X.
\end{aligned}$$

It is not difficult to verify that this defines an information system whose objects are (the graphs of) all partial functions from A to B.

A final example is provided by any fixed partial functional Φ. A token should now be a pair (φ_0, n) where φ_0 is a finite function and $\Phi(\varphi_0)$ is defined with value n. Thus if we take $\textsc{Con}$ to be the set of all finite sets of tokens and for $X := \{\, (\varphi_i, n_i) \mid i = 1 \dots k \,\}$ define $X \vdash (\varphi_0, n)$ iff φ_0 extends some φ_i, then this structure becomes an information system. The objects in this case are all sets x of tokens with the property that whenever (φ_0, n) belongs to x, then also all (φ_0', n) with φ_0' extending φ_0 belong to x.

REMARK. Suppose $\mathbf{A} = (A, \textsc{Con}, \vdash)$ is an information system. Then it is easy to see that

1. $\emptyset \in \textsc{Con}$;

2. If $X \in \textsc{Con}$, $Y \subseteq X$ and $Y \vdash a$, then $X \vdash a$;

3. If $X \in \textsc{Con}$ and $Y \subseteq^{\text{fin}} A$ and $X \vdash a$ for every $a \in Y$, then $X \cup Y \in \textsc{Con}$ by axiom 3; hence $Y \in \textsc{Con}$ by axiom 1.

4. The intersection of any number of ideals is an ideal again.

4.4. *Function spaces.* We now define the 'function space' $\mathbf{A} \to \mathbf{B}$ between two information systems $\mathbf{A}$ and $\mathbf{B}$.

Definition. Let $\mathbf{A} = (A, \textsc{Con}_A, \vdash_A)$ and $\mathbf{B} = (B, \textsc{Con}_B, \vdash_B)$ be information systems. Then the structure $\mathbf{A} \to \mathbf{B} = (C, \textsc{Con}, \vdash)$ is defined as follows:

1. The set C of tokens is $C = \textsc{Con}_A \times B$.

2. The set $\textsc{Con}$ consists of of all finite sets $\{(X_1, b_1), \dots, (X_n, b_n)\} \subseteq C$ such that for every index set $I \subseteq \{1, \dots, n\}$,
$$\bigcup_{i \in I} X_i \in \textsc{Con}_A \implies \{\, b_i \mid i \in I \,\} \in \textsc{Con}_B.$$

3. For the definition of the entailment relation $\vdash$ it is helpful first to define the notion of an *application* of $W := \{(X_1, b_1), \dots, (X_n, b_n)\} \in \textsc{Con}$ to $X \in \textsc{Con}_A$:
$$WX = \{(X_1, b_1), \dots, (X_n, b_n)\}X := \{\, b_i \mid X \vdash_A X_i \,\}.$$
From the definition of $\textsc{Con}$ we know that this set is in $\textsc{Con}_B$. Clearly application is *monotone in the second argument*, in the sense that $X \vdash_A X'$ implies $WX' \subseteq WX$, hence $WX \vdash_B WX'$. Now define $W \vdash (X, b)$ by
$$W \vdash (X, b) \quad :\equiv \quad WX \vdash_B b.$$

In fact, application is also *monotone in the first argument*, i.e. $W \vdash W'$ implies $WX \vdash_B W'X$. To see this let $W' = \{(X_1, b_1), \dots, (X_n, b_n)\}$ and observe that
$$WX \vdash_B \bigcup \{\, WX_i \mid X \vdash_A X_i \,\} \vdash_B \{\, b_i \mid X \vdash_A X_i \,\} = W'X.$$

Lemma. *If $\mathbf{A}$ and $\mathbf{B}$ are information systems, then so is $\mathbf{A} \to \mathbf{B}$ defined as above.*

Proof. Let $\mathbf{A} = (A, \textsc{Con}_A, \vdash_A)$ and $\mathbf{B} = (B, \textsc{Con}_B, \vdash_B)$. The axioms 1, 2 and 4 are clearly satisfied. For axiom 3, suppose
$$\{(X_1, b_1), \dots, (X_n, b_n)\} \vdash (X, b), \quad \text{i.e.} \quad \{\, b_j \mid X \vdash_A X_j \,\} \vdash_B b.$$

We have to show that $\{(X_1, b_1), \dots, (X_n, b_n), (X, b)\} \in \textsc{Con}$. So let $I \subseteq \{1, \dots, n\}$ and suppose
$$X \cup \bigcup_{i \in I} X_i \in \textsc{Con}_A.$$

We must show that $b \cup \{ b_i \mid i \in I \} \in \mathrm{Con}_B$. Let $J \subseteq \{1, \dots, n\}$ consist of those j with $X \vdash_A X_j$. Then also

$$X \cup \bigcup_{i \in I} X_i \cup \bigcup_{j \in J} X_j \in \mathrm{Con}_A.$$

Since

$$\bigcup_{i \in I} X_i \cup \bigcup_{j \in J} X_j \in \mathrm{Con}_A,$$

from the consistency of $\{(X_1, b_1), \dots, (X_n, b_n)\}$ we can conclude that

$$\{ b_i \mid i \in I \} \cup \{ b_j \mid j \in J \} \in \mathrm{Con}_B.$$

But $\{ b_j \mid j \in J \} \vdash_B b$ by assumption. Hence

$$\{ b_i \mid i \in I \} \cup \{ b_j \mid j \in J \} \cup \{b\} \in \mathrm{Con}_B.$$

For axiom 5, suppose

$$W \vdash \{(X_1, b_1), \dots, (X_n, b_n)\} \quad \text{and} \quad \{(X_1, b_1), \dots, (X_n, b_n)\} \vdash (X, b).$$

We have to show that $WX \vdash_B b$. Note that from $X \vdash_A X_i$ we can conclude $WX \vdash_B WX_i$ by the monotonicity of application in the second argument. Hence

$$WX \vdash_B \bigcup \{ WX_i \mid X \vdash_A X_i \} \vdash_B \{ b_i \mid X \vdash_A X_i \} \vdash_B b. \qquad \square$$

We shall now give two alternative characterizations of the function space: firstly as 'approximable maps', and secondly as continuous maps w.r.t. the so-called Scott topology.

4.5. *Approximable Maps.* We want to study 'information respecting' maps from $\mathbf{A}$ into $\mathbf{B}$. Such a map is given by a relation r between Con_A and B, where $r(X, b)$ intuitively means that whenever we are given the information $X \in \mathrm{Con}_A$, then we know that at least the token b appears in the value.

Definition. Let $\mathbf{A} = (A, \mathrm{Con}_A, \vdash_A)$ and $\mathbf{B} = (B, \mathrm{Con}_B, \vdash_B)$ be information systems. A relation $r \subseteq \mathrm{Con}_A \times B$ is an *approximable map* if it satisfies

1. If $r(X, b_1), \dots, r(X, b_n)$, then $\{b_1, \dots, b_n\} \in \mathrm{Con}_B$;
2. If $r(X, b_1), \dots, r(X, b_n)$ and $\{b_1, \dots, b_n\} \vdash_B b$, then $r(X, b)$;
3. If $r(X', b)$ and $X \vdash_A X'$, then $r(X, b)$.

We write $r \colon \mathbf{A} \to \mathbf{B}$ to mean r is an approximable map from $\mathbf{A}$ to $\mathbf{B}$.

THEOREM. *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be information systems. Then the objects of* $\mathbf{A} \to \mathbf{B}$ *are exactly the approximable maps from* $\mathbf{A}$ *to* $\mathbf{B}$.

PROOF. Let $\mathbf{A} = (A, \mathrm{Con}_A, \vdash_A)$ and $\mathbf{B} = (B, \mathrm{Con}_B, \vdash_B)$. If $r \in |\mathbf{A} \to \mathbf{B}|$ then $r \subseteq \mathrm{Con}_A \times B$ is consistent and deductively closed. We have to show that r satisfies the axioms for approximable maps.

1. Let $r(X, b_1), \dots, r(X, b_n)$. We must show that $\{b_1, \dots, b_n\} \in \mathrm{Con}_B$. But this clearly follows from the consistency of r.

2. Let $r(X, b_1), \dots, r(X, b_n)$ and $\{b_1, \dots, b_n\} \vdash_B b$. We must show that $r(X, b)$. But

$$\{(X, b_1), \dots, (X, b_n)\} \vdash (X, b)$$

by the definition of the entailment relation $\vdash$ in $\mathbf{A} \to \mathbf{B}$, hence $r(X, b)$ by the deductive closure of r.

3. Let $X \vdash_A X'$ and $r(X', b)$. We must show that $r(X, b)$. But

$$\{(X', b)\} \vdash (X, b)$$

since $\{(X', b)\}X = \{b\}$ (which follows from $X \vdash_A X'$), hence again $r(X, b)$ by the deductive closure of r.

For the other direction suppose that $r \colon \mathbf{A} \to \mathbf{B}$ is an approximable map. We must show that $r \in |\mathbf{A} \to \mathbf{B}|$.

Consistency of r: suppose $r(X_1, b_1), \dots, r(X_n, b_n)$ and $X = \bigcup\{\, X_i \mid i \in I \,\} \in \mathrm{Con}_A$ for some $I \subseteq \{1, \dots, n\}$. We must show that $\{\, b_i \mid i \in I \,\} \in \mathrm{Con}_B$. Now from $r(X_i, b_i)$ and $X \vdash_A X_i$ we obtain $r(X, b_i)$ by axiom 3 for any $i \in I$, and hence $\{\, b_i \mid i \in I \,\} \in \mathrm{Con}_B$ by axiom 1.

Deductive closure of r: suppose $r(X_{,1} b_1), \dots, r(X_n, b_n)$ and

$$W := \{(X_1, b_1), \dots, (X_n, b_n)\} \vdash (X, b).$$

We must show $r(X, b)$. By definition of $\vdash$ for $\mathbf{A} \to \mathbf{B}$ we have $WX \vdash_B b$, which is $\{\, b_i \mid X \vdash_A X_i \,\} \vdash_B b$. Further by our assumption $r(X_i, b_i)$ we know $r(X, b_i)$ by axiom 3 for all i with $X \vdash_A X_i$. Hence $r(X, b)$ by axiom 2. □

4.6. *Continuous Maps.* We now introduce the Scott topology.

DEFINITION. Suppose $\mathbf{A} = (A, \mathrm{Con}, \vdash)$ is an information system and $X \in \mathrm{Con}$. Define $\overline{X}$, the *deductive closure* of X, by

$$\overline{X} := \{\, a \in A \mid X \vdash a \,\}.$$

Define also $U_X \subseteq |\mathbf{A}|$ by

$$U_X := \{\, x \in |\mathbf{A}| \mid X \subseteq^{\mathrm{fin}} x \,\}.$$

Note that, since the objects $x \in |\mathbf{A}|$ are deductively closed, if $x \in U_X$ then $\overline{X} \subseteq x$.

LEMMA. *The system of all U_X with $X \in \mathrm{Con}$ forms the basis of a topology on $|\mathbf{A}|$, called the Scott topology.*

PROOF. Suppose $X, Y \in \mathrm{Con}$ and $x \in U_X \cap U_Y$. We have to find $Z \in \mathrm{Con}$ such that $x \in U_Z \subseteq U_X \cap U_Y$. Choose $Z = X \cup Y$. $\square$

LEMMA. *Let $\mathbf{A}$ be an information system and $U \subseteq |\mathbf{A}|$. Then the following are equivalent.*

1. *U is open in the Scott topology.*
2. *U satisfies*
   - *If $x \in U$ and $x \subseteq y$, then $y \in U$ (Alexandrov condition).*
   - *If $x \in U$ then $\overline{X} \in U$ for some $X \subseteq^{\mathrm{fin}} x$ (Scott condition).*
3. *$U = \bigcup_{\overline{X} \in U} U_X$.*

So open sets U may be seen as those determined by a (possibly infinite) system of finitely observable properties, namely all X such that $\overline{X} \in U$.

PROOF. $1 \Longrightarrow 2$. If U is open, then U is the union of some U_X's, $X \in \mathrm{Con}$. Since each U_X is upward closed, also U is; this proves the Alexandrov condition. For the Scott condition assume $x \in U$. Then $x \in U_X \subseteq U$ for some $X \in \mathrm{Con}$. Hence $X \subseteq^{\mathrm{fin}} x$, and $\overline{X} \in U$ since $\overline{X} \in U_X$.

$2 \Longrightarrow 3$. Assume that $U \subseteq |\mathbf{A}|$ satisfies the Alexandrov and Scott conditions. Let $x \in U$. By the Scott condition, $\overline{X} \in U$ for some $X \subseteq^{\mathrm{fin}} x$, so $x \in U_X$ for this X. Conversely, let $x \in U_X$ for some $\overline{X} \in U$. Then $\overline{X} \subseteq x$. Now $x \in U$ follows from $\overline{X} \in U$ by the Alexandrov condition.

$3 \Longrightarrow 1$. The U_X's are the basic open sets of the Scott topology. $\square$

We now give some simple characterizations of the continuous functions $f\colon |\mathbf{A}| \to |\mathbf{B}|$. Call f *monotone* if $x \subseteq y$ implies $f(x) \subseteq f(y)$. Call $D \subseteq |\mathbf{A}|$ *directed* if for any $x, y \in D$ there is a $z \in D$ such that $x \subseteq z$ and $y \subseteq z$; note that then $\bigcup D \in |\mathbf{A}|$. An important example of a directed set is the set $\{\, \overline{X} \mid X \subseteq^{\mathrm{fin}} x \,\}$ of *finitely generated* or *compact* approximations (f.g. approximations for short) of a given $x \in |\mathbf{A}|$. Any $x \in |\mathbf{A}|$ can be written as the union of its finitely generated approximations:

$$x = \bigcup \{\, \overline{X} \mid X \subseteq^{\mathrm{fin}} x \,\}.$$

PROPOSITION. *Let $\mathbf{A}$ and $\mathbf{B}$ be information systems and $f\colon |\mathbf{A}| \to |\mathbf{B}|$. Then the following are equivalent.*

1. *f is continuous w.r.t. the Scott topology*
2. *f is monotone and satisfies the 'Principle of Finite Support' PFS: If $b \in f(x)$, then $b \in f(\overline{X})$ for some $X \subseteq^{\mathrm{fin}} x$.*

3. f is monotone and commutes with directed unions: for every directed $D \subseteq |\mathbf{A}|$

$$f(\bigcup_{x \in D} x) = \bigcup_{x \in D} f(x).$$

Note that in 3 the set $\{ f(x) \mid x \in D \}$ is directed by monotonicity of f, hence its union is indeed an object in $|\mathbf{B}|$. Note also that from PFS it follows immediately that if $Y \subseteq^{\mathrm{fin}} f(x)$, then $Y \subseteq^{\mathrm{fin}} f(\overline{X})$ for some $X \subseteq^{\mathrm{fin}} x$.

So continuous maps $f \colon |\mathbf{A}| \to |\mathbf{B}|$ are those that can be completely described from the point of view of finite approximations of the abstract objects $x \in |\mathbf{A}|$ and $f(x) \in |\mathbf{B}|$: Whenever we are given a finite approximation Y to the value $f(x)$, then there is a finite approximation X to the argument x such that already $f(\overline{X})$ contains the information in Y; note that by monotonicity $f(\overline{X}) \subseteq f(x)$.

Proof. $1 \Longrightarrow 2$. Let f be continuous. Then for any basic open set $U_Y \in |\mathbf{B}|$ (so $Y \in \mathrm{Con}_B$) the set $f^{-1}[U_Y] = \{ x \mid Y \subseteq^{\mathrm{fin}} f(x) \}$ is open in $|\mathbf{A}|$. To prove monotonicity assume $x \subseteq y$; we must show $f(x) \subseteq f(y)$. So let $b \in f(x)$, i.e. $\{b\} \subseteq^{\mathrm{fin}} f(x)$. The open set $\{ z \mid \{b\} \subseteq^{\mathrm{fin}} f(z) \}$ satisfies the Alexandrov condition, so from $x \subseteq y$ we can infer $\{b\} \subseteq^{\mathrm{fin}} f(y)$, i.e. $b \in f(y)$. To prove PFS assume $b \in f(x)$. The open set $\{ z \mid \{b\} \subseteq^{\mathrm{fin}} f(z) \}$ satisfies the Scott condition, so for some $X \subseteq^{\mathrm{fin}} x$ we have $\{b\} \subseteq^{\mathrm{fin}} f(\overline{X})$.

$2 \Longrightarrow 1$. Assume that f satisfies monotonicity and PFS. We must show that f is continuous, i.e. that for any fixed $Y \in \mathrm{Con}_B$ the set $f^{-1}[U_Y] = \{ x \mid Y \subseteq^{\mathrm{fin}} f(x) \}$ is open. We prove

$$\{ x \mid Y \subseteq^{\mathrm{fin}} f(x) \} = \bigcup \{ U_X \mid X \in \mathrm{Con}_A \text{ and } Y \subseteq^{\mathrm{fin}} f(\overline{X}) \}.$$

Let $Y \subseteq^{\mathrm{fin}} f(x)$. Then by PFS $Y \subseteq^{\mathrm{fin}} f(\overline{X})$ for some $X \in \mathrm{Con}_A$ such that $X \subseteq^{\mathrm{fin}} x$, and $X \subseteq^{\mathrm{fin}} x$ implies $x \in U_X$. Conversely, let $x \in U_X$ for some $X \in \mathrm{Con}_A$ such that $Y \subseteq^{\mathrm{fin}} f(\overline{X})$. Then $\overline{X} \subseteq x$, hence $Y \subseteq^{\mathrm{fin}} f(x)$ by monotonicity.

For $2 \Longleftrightarrow 3$ assume that f is monotone. Let f satisfy PFS, and $D \subseteq |\mathbf{A}|$ be directed. $f(\bigcup_{x \in D} x) \supseteq \bigcup_{x \in D} f(x)$ follows from monotonicity. For the reverse inclusion let $b \in f(\bigcup_{x \in D} x)$. Then by PFS $b \in f(\overline{X})$ for some $X \subseteq^{\mathrm{fin}} \bigcup_{x \in D} x$. From the directedness and the fact that X is finite we obtain $X \subseteq^{\mathrm{fin}} z$ for some $z \in D$. From $b \in f(\overline{X})$ and monotonicity we infer $b \in f(z)$. Conversely, let f commute with directed unions, and assume $b \in f(x)$. Then

$$b \in f(x) = f(\bigcup_{X \subseteq^{\mathrm{fin}} x} \overline{X}) = \bigcup_{X \subseteq^{\mathrm{fin}} x} f(\overline{X}),$$

so $b \in f(\overline{X})$ for some $X \subseteq^{\mathrm{fin}} x$. □

Clearly the identity and constant functions are continuous, and also the *composition* $g \circ f$ of continuous functions $f \colon |\mathbf{A}| \to |\mathbf{B}|$ and $g \colon |\mathbf{B}| \to |\mathbf{C}|$.

THEOREM. *Let* $\mathbf{A}$ *and* $\mathbf{B} = (B, \mathrm{Con}_B, \vdash_B)$ *be information systems. Then the objects of* $\mathbf{A} \to \mathbf{B}$ *are in a natural bijective correspondence with the continuous functions from* $|\mathbf{A}|$ *to* $|\mathbf{B}|$*, as follows.*

- *With any approximable map* $r\colon \mathbf{A} \to \mathbf{B}$ *we can associate a continuous function* $|r|\colon |\mathbf{A}| \to |\mathbf{B}|$ *by*
$$|r|(z) := \{\, b \in B \mid r(X, b) \textit{ for some } X \subseteq^{\mathrm{fin}} z \,\}.$$
- *Conversely, with any continuous function* $f\colon |\mathbf{A}| \to |\mathbf{B}|$ *we can associate an approximable map* $\hat{f}\colon \mathbf{A} \to \mathbf{B}$ *by*
$$\hat{f}(X, b) \quad :\equiv \quad b \in f(\overline{X}).$$

These assignments are inverse to each other, i.e. $f = |\hat{f}|$ *and* $r = \widehat{|r|}$*.*

PROOF. Let r be an object of $\mathbf{A} \to \mathbf{B}$; then by the theorem just proved r is an approximable map. We first show that $|r|$ is well-defined. So let $z \in |\mathbf{A}|$.

$|r|(z)$ is consistent: let $b_1, \dots, b_n \in |r|(z)$. Then there are $X_1, \dots, X_n \subseteq^{\mathrm{fin}} z$ such that $r(X_i, b_i)$. Hence $X := X_1 \cup \ldots \cup X_n \subseteq^{\mathrm{fin}} z$ and $r(X, b_i)$ by axiom 3 of approximable maps. Now from axiom 1 we can conclude that $\{b_1, \dots, b_n\} \in \mathrm{Con}_B$.

$|r|(z)$ is deductively closed: let $b_1, \dots, b_n \in |r|(z)$ and $\{b_1, \dots, b_n\} \vdash_B b$. We must show $b \in |r|(z)$. As before we find $X \subseteq^{\mathrm{fin}} z$ such that $r(X, b_i)$. Now from axiom 2 we can conclude $r(X, b)$ and hence $b \in |r|(z)$.

To prove continuity of $|r|$ let $Y \in \mathrm{Con}_B$; we must show that $|r|^{-1}[U_Y]$ is open. Now for every $z \in |\mathbf{A}|$

$$\begin{aligned}
z \in |r|^{-1}[U_Y] \quad &\equiv \quad |r|(z) = \{\, b \in B \mid r(X, b) \text{ for some } X \subseteq^{\mathrm{fin}} z \,\} \in U_Y \\
&\equiv \quad Y \subseteq^{\mathrm{fin}} \{\, b \in B \mid r(X, b) \text{ for some } X \subseteq^{\mathrm{fin}} z \,\} \\
&\equiv \quad \forall b {\in} Y \exists X \, [X \subseteq^{\mathrm{fin}} z \wedge r(X, b)] \\
&\equiv \quad \forall b {\in} Y \exists X \, [z \in U_X \wedge r(X, b)] \\
&\equiv \quad z \in \bigcap_{b \in Y} \bigcup \{\, U_X \mid r(X, b) \,\}
\end{aligned}$$

Since Y is finite, this implies that $|r|^{-1}[U_Y]$ is open.

Now let $f\colon |\mathbf{A}| \to |\mathbf{B}|$ be continuous. It is easy to verify that $\hat{f}$ is indeed an approximable map. Furthermore

$$\begin{aligned}
b \in |\hat{f}|(z) \quad &\equiv \quad \hat{f}(X, b) \text{ for some } X \subseteq^{\mathrm{fin}} z \\
&\equiv \quad b \in f(\overline{X}) \text{ for some } X \subseteq^{\mathrm{fin}} z \\
&\equiv \quad b \in f(z) \text{ by monotonicity and PFS.}
\end{aligned}$$

Finally, for any approximable map $r\colon \mathbf{A} \to \mathbf{B}$ we have

$$\begin{aligned}
r(X, b) \quad &\equiv \quad \exists Y {\subseteq^{\mathrm{fin}}} \overline{X} \, r(Y, b) \quad \text{by axiom 3 for approximable maps} \\
&\equiv \quad b \in |r|(\overline{X}) \\
&\equiv \quad \widehat{|r|}(X, b)
\end{aligned}$$

This completes the proof. $\square$

REMARK. From now on we will usually write $r(z)$ for $|r|(z)$, and similarly $f(X, b)$ for $\hat{f}(X, b)$. It should always be clear from the context where the mods and hats should be inserted.

4.7. *Cartesian Products.* We define an information system $\mathbf{A} \times \mathbf{B} = (C, \mathrm{Con}, \vdash)$ from $\mathbf{A}$ and $\mathbf{B}$ in such a way that the inclusion ordering $\subseteq$ on the objects of $|\mathbf{A} \times \mathbf{B}|$ is isomorphic to the cartesian product of the inclusion orderings on $|\mathbf{A}|$ and $|\mathbf{B}|$. Without loss of generality we may assume that A and B are disjoint. But then any pair (x, y) of elements $x \in |\mathbf{A}|$ and $y \in |\mathbf{B}|$ can be approximated in each component separately. Hence we choose as tokens in $\mathbf{A} \times \mathbf{B}$ simply the union $C := A \cup B$. Consistency and entailment is inherited in the expected way from $\mathbf{A}$ and $\mathbf{B}$:

$$\begin{aligned} X \in \mathrm{Con} \quad &:\equiv \quad X \cap A \in \mathrm{Con}_A \text{ and } X \cap B \in \mathrm{Con}_B, \\ X \vdash c \quad &:\equiv \quad (c \in A \implies X \cap A \vdash_A c) \text{ and } (c \in B \implies X \cap B \vdash_B c). \end{aligned}$$

It is then obvious that we have

LEMMA. *If $\mathbf{A}$ and $\mathbf{B}$ are information systems with $A \cap B = \emptyset$, then so is $\mathbf{A} \times \mathbf{B}$ defined as above. The objects of $\mathbf{A} \times \mathbf{B}$ are exactly the unions $x \cup y$ of the objects of $x \in |\mathbf{A}|$ and of $y \in |\mathbf{B}|$.*

When using both $\to$ and $\times$ to build information systems, we assume that $\times$ has a higher precedence that $\to$, hence e.g. $\mathbf{A} \times \mathbf{B} \to \mathbf{C}$ means $(\mathbf{A} \times \mathbf{B}) \to \mathbf{C}$.

REMARK. Clearly the pairs $(x, y) \in |\mathbf{A}| \times |\mathbf{B}|$ are in a natural bijective correspondence with the objects $x \cup y$ of $|\mathbf{A} \times \mathbf{B}|$. Therefore from now on we will usually write x, y for $x \cup y$, to increase readability. It should be clear from the context where a comma needs to be replaced by a union.

Clearly the *projections* $\pi_i \colon \mathbf{A}_1 \times \mathbf{A}_2 \to \mathbf{A}_i$ defined by $\pi_i(x_1, x_2) := x_i$ for $i = 1, 2$ are continuous.

LEMMA. *(Universal property of cartesian products). Let $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ be information systems such that $A \cap B = \emptyset$. Then for any pair $f \colon \mathbf{C} \to \mathbf{A}$ and $g \colon \mathbf{C} \to \mathbf{B}$ of continuous functions there is a unique continuous function $h \colon \mathbf{C} \to \mathbf{A} \times \mathbf{B}$ such that $f = \pi_1 \circ h$ and $g = \pi_1 \circ h$.*

$$\begin{tikzcd} & \mathbf{A} \\ \mathbf{C} \arrow[ur, "f"] \arrow[r, "h"] \arrow[dr, "g"'] & \mathbf{A} \times \mathbf{B} \arrow[u, "\pi_1"'] \arrow[d, "\pi_2"] \\ & \mathbf{B} \end{tikzcd}$$

Proof. Uniqueness follows from the fact that any such h must satisfy $h(x) = (f(x), g(x))$ for $x \in |\mathbf{C}|$. To prove existence, define $h(x) := (f(x), g(x))$ for $x \in |\mathbf{C}|$. We must show that this h is continuous. Monotonicity is obvious; for PFS assume $b \in (f(x), g(x))$. Since $A \cap B = \emptyset$, the token b must be in exactly one component, say B. So $b \in g(x)$, hence $b \in g(\overline{X})$ for some $X \subseteq^{\text{fin}} x$ and therefore $b \in h(\overline{X}) = (f(\overline{X}), g(\overline{X}))$. □

As an application let us construct the *product* $f \times g$ of two continuous functions $f: \mathbf{A} \to \mathbf{C}$ and $g: \mathbf{B} \to \mathbf{D}$ (with $A \cap B = \emptyset$).

$$\begin{tikzcd}
 & & \mathbf{C} \\
\mathbf{A} \times \mathbf{B} \arrow[urr, "f \circ \pi_1"] \arrow[rr, "f \times g"] \arrow[drr, "g \circ \pi_2"'] & & \mathbf{C} \times \mathbf{D} \arrow[u, "\pi_1"'] \arrow[d, "\pi_2"] \\
 & & \mathbf{D}
\end{tikzcd}$$

$f \times g$ satisfies $(f \times g)(x, y) = (f(x), g(y))$ for $x \in |\mathbf{A}|$ and $y \in |\mathbf{B}|$.

Lemma. *Let $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ be information systems with $A \cap B = \emptyset$. A function $f: \mathbf{A} \times \mathbf{B} \to \mathbf{C}$ is continuous iff it is continuous in each argument separately, i.e. all its sections $f_1^y: \mathbf{A} \to \mathbf{C}$, $f_1^y(x) := f(x, y)$ for fixed $y \in |\mathbf{B}|$ and similarly $f_2^x: \mathbf{B} \to \mathbf{C}$, $f_2^x(y) := f(x, y)$ for fixed $x \in |\mathbf{A}|$ are continuous.*

Proof. The function sending x to (x, y) for fixed y clearly is continuous, hence by composition so is f_1^y; for f_2^x the argument is similar. Conversely, assume that all f_1^y and f_2^x are continuous. To prove monotonicity of f, assume $x \subseteq x'$ and $y \subseteq y'$ with $x, x' \in |\mathbf{A}|$ and $y, y' \in |\mathbf{B}|$. Then $f(x, y) \subseteq f(x', y) \subseteq f(x', y')$ by monotonicity of the sections. To prove PFS for f, assume $c \in f(x, y)$. Then $c \in f_1^x(y)$, hence

$$c \in f_1^x(\overline{Y}) = f(x, \overline{Y}) = f_2^{\overline{Y}}(x)$$

for some $Y \subseteq^{\text{fin}} y$ by PFS for f_1^x, and also

$$c \in f_2^{\overline{Y}}(\overline{X}) = f(\overline{X}, \overline{Y}) = f(\overline{X \cup Y})$$

for some $X \subseteq^{\text{fin}} x$ by PFS for $f_2^{\overline{Y}}$. □

4.8. *Evaluation and Currying.* We now show that the information systems together with the approximable mappings form a 'cartesian closed category'.

LEMMA. *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be information systems. Then the function*

$$\begin{aligned}&\mathrm{eval}\colon (\mathbf{A} \to \mathbf{B}) \times \mathbf{A} \to \mathbf{B}\\ &\mathrm{eval}(f, x) := f(x) \quad \textit{for } f \in |\mathbf{A} \to \mathbf{B}|,\ x \in |\mathbf{A}|\end{aligned}$$

is continuous.

PROOF. To avoid confusion we shall use the more appropriate (but less suggestive) letter r instead of f, and also use the correct notation $|r|(x)$ instead of $r(x)$.

By the previous lemma it suffices to prove continuity in each component separately. For the second component this is obvious. For the first component monotonicity follows from the definition of $|r|$. To prove PFS assume $b \in \mathrm{eval}(r, x) = |r|(x)$. Then $r(X, b)$ for some $X \subseteq^{\mathrm{fin}} x$, hence for $W := \{(X, b)\}$ we have $W \subseteq^{\mathrm{fin}} r$ and

$$\mathrm{eval}(\overline{W}, x) = |\overline{W}|(x) = \{\, c \in B \mid \overline{W}(Y, c) \text{ for some } Y \subseteq^{\mathrm{fin}} x \,\} \ni b. \qquad \square$$

LEMMA. *Let* $\mathbf{A}$, $\mathbf{B}$ *and* $\mathbf{C}$ *be information systems with* $A \cap B = \emptyset$. *Then the function*

$$\begin{aligned}&\mathrm{curry}\colon (\mathbf{A} \times \mathbf{B} \to \mathbf{C}) \to (\mathbf{A} \to (\mathbf{B} \to \mathbf{C}))\\ &\mathrm{curry}(f)(x)(y) := f(x, y) \quad \textit{for } f \in |\mathbf{A} \times \mathbf{B}|,\ x \in |\mathbf{A}| \textit{ and } y \in |\mathbf{B}|\end{aligned}$$

is well-defined and continuous.

PROOF. We again use the letter r instead of f, and also use the correct notation $|r|(x)$ instead of $r(x)$.

Fix $r \in |\mathbf{A} \times \mathbf{B} \to \mathbf{C}|$. Then for fixed $x \in |\mathbf{A}|$ the function $k := |r|_2^x$ sending $y \in |\mathbf{B}|$ to $|r|(x, y) \in |\mathbf{C}|$ is continuous, since it is a section of $|r|$. We now show that the function h sending $x \in |\mathbf{A}|$ into $\hat{k} \in |\mathbf{B} \to \mathbf{C}|$ is continuous. For monotonicity assume $x \subseteq x'$. Let $\hat{k}' = h(x')$. We must show $\hat{k} \subseteq \hat{k}'$. It suffices to show that $\hat{k}(Y, c)$ implies $\hat{k}'(Y, c)$, i.e. $c \in k(\overline{Y})$ implies $c \in k'(\overline{Y})$. But $k(\overline{Y}) \subseteq k'(\overline{Y})$ by monotonicity of $|r|$. For PFS assume $(Y, c) \in h(x) = \hat{k}$. We want $X \subseteq^{\mathrm{fin}} x$ such that $(Y, c) \in h(\overline{X})$. Now $(Y, c) \in \hat{k}$, so $c \in k(\overline{Y}) = |r|(x, \overline{Y})$. PFS for the section $|r|_1^{\overline{Y}}$ yields an $X \subseteq^{\mathrm{fin}} x$ such that $c \in |r|(\overline{X}, \overline{Y}) = |h(\overline{X})|(\overline{Y})$, i.e. $(Y, c) \in h(\overline{X})$.

It remains to show that the function g sending r into $\hat{h}$ is continuous. Monotonicity follows from the definition of $|r|$. For PFS assume $(X, (Y, c)) \in g(r)$, for some $X \in \mathrm{CON}_A$, $Y \in \mathrm{CON}_B$ and $c \in C$. Then it is easy to verify that $(X, (Y, c)) \in g(\overline{W})$ for $W := \{(X \cup Y, c)\} \subseteq^{\mathrm{fin}} r$. $\square$

Another way to say this is that for any continuous $f\colon \mathbf{A} \times \mathbf{B} \to \mathbf{C}$ there is a continuous $c_f\colon \mathbf{A} \to (\mathbf{B} \to \mathbf{C})$ such that $\mathrm{eval} \circ (c_f \times \mathrm{id}) = f$:

$$\begin{tikzcd}
(\mathbf{B} \to \mathbf{C}) \times \mathbf{B} \arrow[r, "\mathrm{eval}"] & \mathbf{C} \\
\mathbf{A} \times \mathbf{B} \arrow[u, "c_f \times \mathrm{id}"] \arrow[ur, "f"'] &
\end{tikzcd}$$

Moreover, such a c_f clearly is uniquely determined. This together with the fact that we can form cartesian products $\mathbf{A} \times \mathbf{B}$ (satisfying the universal property above) and the existence of a 'terminal' information system $(\emptyset, \{\emptyset\}, \emptyset)$ (such that for every $\mathbf{A}$ there is a unique $f\colon \mathbf{A} \to (\emptyset, \{\emptyset\}, \emptyset))$, establishes that the information systems with continuous maps between them form a *cartesian closed category*.

4.9. *Typed Lambda Terms.* An important consequence of working in a cartesian closed category is that every 'explicit definition' involving higher types actually defines an object. We first explain precisely what makes up such an explicit definition.

(*Finite* or *simple*) *types* $\rho, \sigma, \tau \dots$ are built from a symbol ι for the type of natural numbers by *function type* formation $\rho \to \sigma$ and *product type* formation $\rho \times \sigma$. The *level* $\mathrm{lev}(\rho)$ of a type ρ is defined inductively by

$$\begin{aligned} \mathrm{lev}(\iota) &:= 0 \\ \mathrm{lev}(\rho \to \sigma) &:= \max(\mathrm{lev}(\rho) + 1, \mathrm{lev}(\sigma)) \\ \mathrm{lev}(\rho \times \sigma) &:= \max(\mathrm{lev}(\rho), \mathrm{lev}(\sigma)). \end{aligned}$$

The type constructor $\to$ is understood to be associative to the right, so $\rho_1 \to \dots \to \rho_{n-1} \to \rho_n$ means $(\rho_1 \to \dots \to (\rho_{n-1} \to \rho_n) \dots)$. Similarly $\times$ is understood to be associative to the left. We also let $\times$ have a higher precedence than $\to$, so $\rho \times \sigma \to \tau$ means $(\rho \times \sigma) \to \tau$. Note that any type without $\times$ can be written uniquely in the form $\rho_1 \to \dots \to \rho_n \to \iota$ and its level is $\max(\mathrm{lev}(\rho_1), \dots, \mathrm{lev}(\rho_n)) + 1$ if $n > 0$, and 0 otherwise. For simplicity we assume that there is only one ground type ι, the type of natural numbers.

Typed λ-terms are built up from typed variables $x^\rho, y^\rho, z^\rho, \dots$ (countably many of each type) and constants denoted by c^ρ by means of the operations of *application* $(M^{\rho\to\sigma} N^\rho)^\sigma$ and *λ-abstraction* $(\lambda x^\rho M^\sigma)^{\rho\to\sigma}$. The type superscripts will be omitted whenever the typing is clear from the context or else immaterial. The variable x in $\lambda x M$ is considered to be bound; with this in mind the set $\mathrm{FV}(M)$ of variables free in M is defined in the expected way.

Definition. We define the information system $\mathbf{C}_\rho = (C_\rho, \mathrm{Con}_\rho, \vdash_\rho)$ by induction on the type ρ:

$$\begin{aligned} &\mathbf{C}_\iota := \mathbb{N} \quad \text{(viewed as a flat information system)} \\ &\mathbf{C}_{\rho\to\sigma} := \mathbf{C}_\rho \to \mathbf{C}_\sigma \\ &\mathbf{C}_{\rho\times\sigma} := \mathbf{C}_\rho \times \mathbf{C}_\sigma. \end{aligned}$$

The *partial continuous functionals* (over $\mathbb{N}$) of type $\rho \to \sigma$ are the continuous functions from $|\mathbf{C}_\rho|$ to $|\mathbf{C}_\sigma|$.

Lemma. *For every λ-term M^ρ and every list $x_1^{\rho_1}, \dots, x_n^{\rho_n}$ of variables containing all variables free in M we have $[\![M]\!]\colon \mathbf{C}_{\rho_1} \times \dots \times \mathbf{C}_{\rho_n} \to \mathbf{C}_\rho$ such that for all*

$u_1 \in |\mathbf{C}_{\rho_1}|, \dots, u_n \in |\mathbf{C}_{\rho_n}|$

$$\begin{aligned} [\![x_i]\!](u_1,\dots,u_n) &= u_i \\ [\![MN]\!](u_1,\dots,u_n) &= [\![M]\!](u_1,\dots,u_n)([\![N]\!](u_1,\dots,u_n)) \\ [\![\lambda x M]\!](u_1,\dots,u_n)(v) &= [\![M]\!](u_1,\dots,u_n,v). \end{aligned}$$

Proof. By induction on the term M. In case M is a variable x_i, let $[\![x_i]\!] := \pi_i^n$, where π_i^n is the projection af an n-tuple to its i-th component. Formally we define π_i^n from the projections π_1, π_2 (introduced above) as follows (recall that $A \times B \times C := (A \times B) \times C$): $\pi_1^1 = \mathrm{id}$, $\pi_{n+1}^{n+1} = \pi_2$, $\pi_i^{n+1} = \pi_i^n \circ \pi_1$ for $i = 1 \dots n$.

For an application term, define $[\![MN]\!] := \mathrm{eval} \circ ([\![M]\!] \times [\![N]\!])$. This is a continuous function by what was proved above, and

$$\begin{aligned} [\![MN]\!](u_1,\dots,u_n) &= \mathrm{eval}([\![M]\!] \times [\![N]\!])(u_1,\dots,u_n) \\ &= \mathrm{eval}([\![M]\!](u_1,\dots,u_n), [\![N]\!](u_1,\dots,u_n)) \\ &= [\![M]\!](u_1,\dots,u_n)([\![N]\!](u_1,\dots,u_n)). \end{aligned}$$

Finally for a λ-abstraction, let $[\![\lambda x M]\!] := \mathrm{curry}([\![M]\!])$. Then by the properties of curry

$$[\![\lambda x M]\!](u_1,\dots,u_n)(v) = \mathrm{curry}([\![M]\!])(u_1,\dots,u_n)(v) = [\![M]\!](u_1,\dots,u_n,v). \qquad \square$$

4.10. *Coherence.* The information systems $\mathbf{C}_\rho$ enjoy the rather pleasant property of *coherence*, which amounts to the possibility to locate inconsistencies in two-element sets of data objects. Generally, an information system $\mathbf{A} = (A, \mathrm{Con}, \vdash)$ is called *coherent* if for any set $X \subseteq^{\mathrm{fin}} A$ we have

$$X \in \mathrm{Con} \equiv \forall a, b {\in} X (\{a, b\} \in \mathrm{Con}).$$

Clearly any flat information system is coherent, and moreover we have

Lemma. *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be information systems.*

1. *If* $\mathbf{B}$ *is coherent, then so is* $\mathbf{A} \to \mathbf{B}$.
2. *If* $\mathbf{A}$ *and* $\mathbf{B}$ *are coherent, then so is* $\mathbf{A} \times \mathbf{B}$.

Proof. Let $\mathbf{A} = (A, \mathrm{Con}_A, \vdash_A)$ and $\mathbf{B} = (B, \mathrm{Con}_B, \vdash_B)$.

1. Let $\mathbf{A} \to \mathbf{B} = (\mathrm{Con}_A \times B, \mathrm{Con}, \vdash)$ and $\{(X_1, b_1), \dots, (X_n, b_n)\} \subseteq \mathrm{Con}_A \times B$. Assume

$$\forall i, j (1 \le i < j \le n \to \{(X_i, b_i), (X_j, b_j)\} \in \mathrm{Con}). \qquad (*)$$

We have to show that $\{(X_1, b_1), \dots, (X_n, b_n)\} \in \mathrm{Con}$. So assume $I \subseteq \{1, \dots, n\}$ and $\bigcup_{i \in I} X_i \in \mathrm{Con}_A$. We must show that $\{\, b_i \mid i \in I \,\} \in \mathrm{Con}_B$. Now since $\mathbf{B}$ is coherent by assumption, it suffices to show that $\{b_i, b_j\} \in \mathrm{Con}_B$ for all $i, j \in I$. So let $i, j \in I$. By assumption we have $X_i \cup X_j \in \mathrm{Con}_A$ and hence by $(*)$ and the definition of Con also $\{b_i, b_j\} \in \mathrm{Con}_B$.

2. Let $\mathbf{A} \times \mathbf{B} = (A \cup B, \textsc{Con}, \vdash)$ (with $A \cap B = \emptyset$). Let $a_1, \ldots, a_n \in A$ and $b_1, \ldots, b_m \in B$ such that any two-element subset of $\{a_1, \ldots, a_n, b_1, \ldots, b_m\}$ is consistent. Then $\{a_1, \ldots, a_n\} \in \textsc{Con}_A$ and $\{b_1, \ldots, b_m\} \in \textsc{Con}_B$ since $\mathbf{A}$ and $\mathbf{B}$ are coherent. Therefore $\{a_1, \ldots, a_n, b_1, \ldots, b_m\} \in \textsc{Con}$ by definition of $\textsc{Con}$. $\square$

COROLLARY. *The information systems $\mathbf{C}_\rho$ are all coherent.* $\square$

5 Computability in higher types

We now use the theory presented in the previous section to set up a framework for computability in our iterated function spaces $\mathbf{C}_\rho$. But first we want want to make it clear that higher types are relevant for our subject of classifying recursive functions: as observed in [55] (cf. also Fortune, Leivant and O'Donnell [16]), the functions F_α, $\alpha < \varepsilon_0$, of the fast growing hierarchy can be defined quite easily from higher type iteration functionals.

As already mentioned in 4.1, a functional is taken to be computable if it is the limit of a computably enumerable set of finite approximations. This is an 'external' notion of computability, and the question arises whether it can be characterized internally, as a class of functionals generated from initial functionals by certain schemata. Such a characterization has been given by Plotkin [46]: a partial continuous functional is computable if and only if it can be defined explicitly from

- fixed point operators introduced into this framework by Platek in his thesis [45],
- the parallel conditional pcond of type $\iota \to \iota \to \iota \to \iota$ and
- a continuous approximation $\exists$ of type $(\iota \to \iota) \to \iota$ to the existential quantifier.

We shall give a detailed proof of this result below.

5.1. *Iteration functionals.* We first extend the definition of the functions F_α into higher types. It is convenient here to introduce *integer types* ρ_n:

$$\begin{aligned} \rho_0 \quad &:= \iota, \\ \rho_{n+1} &:= \rho_n \to \rho_n. \end{aligned}$$

So if $x_0, \ldots, x_{n+1}$ are of integer types $\rho_0, \ldots, \rho_{n+1}$, then we can form $x_{n+1}(x_n)$ (which is of type ρ_n) and hence finally $x_{n+1}(x_n) \ldots (x_0)$; the latter will be abbreviated by $x_{n+1}(x_n, \ldots, x_0)$. Note that $\mathrm{lev}(\rho_n) = n$. Now we define F_α^{n+1} of type ρ_{n+1} for $\alpha < \varepsilon_0$, by the following induction on α.

$$\begin{aligned} F_0^{n+1}(x_n, \ldots, x_0) &:= \begin{cases} 2^{x_0} & \text{if } n = 0 \\ x_n^{(x_0)}(x_{n-1}, \ldots, x_0) & \text{otherwise,} \end{cases} \\ F_{\alpha+1}^{n+1}(x_n, \ldots, x_0) &:= (F_\alpha^{n+1})^{(x_0)}(x_n, \ldots, x_0), \\ F_\lambda^{n+1}(x_n, \ldots, x_0) &:= F_{\lambda[x_0]}^{n+1}(x_n, \ldots, x_0). \end{aligned}$$

Here $x_n^{(y)}(x_{n-1},\dots,x_0)$ denotes $I(y,x_n,\dots,x_0)$ with an iteration functional I of the type $\iota,\rho_n,\rho_{n-1},\dots,\rho_0\to\rho_0$ defined by

$$\begin{aligned} I(0,y,z) &:= z,\\ I(x+1,y,z) &:= y(I(x,y,z)). \end{aligned}$$

In the following lemma, the condition $\beta+\omega^\alpha=\beta\ \#\ \omega^\alpha$ expresses that in the Cantor normal form of β the last summand ω^{β_0} (if it exists) has an exponent $\beta_0\ge\alpha$.

Lemma. *$F_\alpha^{n+1}(F_\beta^n)=F_{\beta+\omega^\alpha}^n$ if $\beta+\omega^\alpha=\beta\ \#\ \omega^\alpha$.*

Proof. We use induction on α. $\alpha=0$:

$$\begin{aligned} F_0^{n+1}(F_\beta^n,x_{n-1},\dots,x_0) &= (F_\beta^n)^{(x_0)}(x_{n-1},\dots,x_0)\\ &= F_{\beta+1}^n(x_{n-1},\dots,x_0) \end{aligned}$$

α successor:

$$\begin{aligned} F_\alpha^{n+1}(F_\beta^n,x_{n-1},\dots,x_0) &= (F_{\alpha-1}^{n+1})^{(x_0)}(F_\beta^n,x_{n-1},\dots,x_0)\\ &= F_{\beta+\omega^{\alpha-1}\cdot x_0}^n(x_{n-1},\dots,x_0) \quad \text{by induction hypothesis}\\ &= F_{(\beta+\omega^\alpha)[x_0]}^n(x_{n-1},\dots,x_0)\\ &= F_{\beta+\omega^\alpha}^n(x_{n-1},\dots,x_0). \end{aligned}$$

α limit:

$$\begin{aligned} F_\alpha^{n+1}(F_\beta^n,x_{n-1},\dots,x_0) &= F_{\alpha[x_0]}^{n+1}(F_\beta^n,x_{n-1},\dots,x_0)\\ &= F_{\beta+\omega^{\alpha[x_0]}}^n(x_{n-1},\dots,x_0) \quad \text{by induction hypothesis}\\ &= F_{(\beta+\omega^\alpha)[x_0]}^n(x_{n-1},\dots,x_0)\\ &= F_{\beta+\omega^\alpha}^n(x_{n-1},\dots,x_0). \end{aligned}$$ □

Hence each F_α^{n+1}, $\alpha<\varepsilon_0$ and therefore in particular the functions $F_\alpha=F_\alpha^1$ for $\alpha<\varepsilon_0$ can be built from the simple functionals F_0^{n+1}, which are essentially iteration functionals, by application alone. Note that in the resulting representation of the functions F_α, $\alpha<\varepsilon_0$, we do not need the fundamental sequences $\lambda[x]$ any more.

5.2. *Abstract computability.* With the preparations done it is now rather straightforward to define computability in our iterated function spaces $\mathbf{C}_\rho$ (based on $\mathbb{N}$ viewed as a flat information system). The tokens and finite sets of tokens are encodable by integers using sequence-coding. It is easy to see that the notions $X\in\mathrm{Con}_\rho$ of consistency and $X\vdash_\rho a$ of entailment correspond to recursive (in fact elementary) relations.

Definition. A partial continuous functional φ of type ρ is said to be *computable* if - when viewed as a set of (codes of) tokens - it is Σ_1^0-definable.

LEMMA. *For all types ρ, σ, τ the functionals*

$$\mathrm{eval}_{\rho,\sigma}\colon (\mathbf{C}_\rho \to \mathbf{C}_\sigma) \times \mathbf{C}_\rho \to \mathbf{C}_\sigma$$
$$\mathrm{curry}_{\rho,\sigma,\tau}\colon (\mathbf{C}_\rho \times \mathbf{C}_\sigma \to \mathbf{C}_\tau) \to (\mathbf{C}_\rho \to (\mathbf{C}_\sigma \to \mathbf{C}_\tau))$$

are computable.

PROOF. The tokens of $\mathrm{eval}_{\rho,\sigma}$ are of the form (W, X, a) with $W \in \mathrm{Con}_{\rho\to\sigma}$, $X \in \mathrm{Con}_\rho$ and $a \in C_\sigma$, and we have

$$(W, X, a) \in \mathrm{eval} \equiv a \in \overline{W}(\overline{X}) \equiv a \in \overline{WX} \equiv WX \vdash a.$$

Here WX is the application of finite approximations introduced in 4.4, and we have made use of $\overline{WX} = \overline{W}(\overline{X})$, which can be proved easily. Thus we have a Σ^0_1-definition of $\mathrm{eval}_{\rho,\sigma}$. For curry we similarly have

$$(W, X, Y, a) \in \mathrm{curry} \equiv a \in \overline{W}(\overline{X \cup Y}) \equiv a \in \overline{W(X \cup Y)} \equiv W(X \cup Y) \vdash a,$$

which again is a Σ^0_1-definition. This completes the proof. □

LEMMA.

1. *If $\varphi\colon \mathbf{C}_\rho \to \mathbf{C}_\sigma$ and $\psi\colon \mathbf{C}_\sigma \to \mathbf{C}_\tau$ are computable, then so is $\psi \circ \varphi$.*
2. *If $\Phi\colon \mathbf{C}_\rho \to \mathbf{C}_\sigma$ and $\varphi \in |\mathbf{C}_\rho|$ are computable, then so is $\Phi(\varphi)$.*
3. *If $\varphi\colon \mathbf{C}_\rho \to \mathbf{C}_\sigma$ and $\psi\colon \mathbf{C}_\rho \to \mathbf{C}_\tau$ are computable, then so is $\varphi \times \psi$.*
4. *The projections $\pi_1\colon \mathbf{C}_\rho \times \mathbf{C}_\sigma \to \mathbf{C}_\rho$ and $\pi_2\colon \mathbf{C}_\rho \times \mathbf{C}_\sigma \to \mathbf{C}_\sigma$ are computable.*

PROOF. 1. The tokens of $\psi \circ \varphi$ are of the form (X, b) with $X \in \mathrm{Con}_\rho$ and $b \in C_\tau$, and we have

$$(X, b) \in \psi \circ \varphi \equiv \exists Y \subseteq^{\mathrm{fin}} \varphi[(Y, b) \in \psi \wedge \forall a{\in}Y\ (X, a) \in \varphi].$$

This is a Σ^0_1-definition, since φ and ψ are computable.

2. Since eval is continuous, we have from PFS

$$a \in \Phi(\varphi) \equiv \exists W \subseteq^{\mathrm{fin}} \Phi \exists X \subseteq^{\mathrm{fin}} \varphi(a \in \overline{W}(\overline{X})) \equiv \exists W \subseteq^{\mathrm{fin}} \Phi \exists X \subseteq^{\mathrm{fin}} \varphi(WX \vdash a).$$

This again is a Σ^0_1 definition.

3 and 4 are proved similarly. □

LEMMA. *For every λ-term M^ρ and every list $x_1^{\rho_1}, \ldots, x_n^{\rho_n}$ of variables containing all variables free in M the functional $[\![M]\!]\colon \mathbf{C}_{\rho_1} \times \ldots \times \mathbf{C}_{\rho_n} \to \mathbf{C}_\rho$ defined above is computable.*

PROOF. The construction of $[\![M]\!]$ (by induction on M) in 4.9 is such that the previous lemma implies immediately its computability. □

5.3. *Least fixed points.* Since the least fixed point operator makes sense for arbitrary information systems, we for a moment re-introduce that level of generality. So let $\mathbf{A}$ be an information system and $f\colon \mathbf{A} \to \mathbf{A}$. An object $x \in |\mathbf{A}|$ is a *fixed point* of f if $f(x) = x$. It is called the least fixed point of f if for any other $y \in |\mathbf{A}|$ with $f(y) = y$ we have $x \subseteq y$.

THEOREM. *(Fixed Point Theorem). Let $\mathbf{A}$ be an information system and $f\colon \mathbf{A} \to \mathbf{A}$. Then f has a least fixed point $Y(f)$ given by*

$$Y(f) := \bigcup_{n\in\mathbb{N}} f^{(n)}(\overline{\emptyset}).$$

Moreover, Y is continuous.

PROOF. From $\overline{\emptyset} \subseteq f(\overline{\emptyset})$ we get $f^{(n)}(\overline{\emptyset}) \subseteq f^{(n+1)}(\overline{\emptyset})$ by induction, since f is monotone. Hence $x := \bigcup_n f^{(n)}(\overline{\emptyset}) \in |\mathbf{A}|$. Moreover, by the characterization of continuity in part 3 of the proposition in 4.6

$$f(x) = f(\bigcup_n f^{(n)}(\overline{\emptyset})\,) = \bigcup_n f^{(n+1)}(\overline{\emptyset}) = x.$$

If y is another fixed point, from $\overline{\emptyset} \subseteq y$ we get $f^{(n)}(\overline{\emptyset}) \subseteq y$ by induction and hence

$$x = \bigcup_n f^{(n)}(\overline{\emptyset}) \subseteq y.$$

Finally we show that Y is continuous. Monotonicity follows immediately from its definition. For PFS assume $b \in Y(f)$, i.e. $b \in f^{(m)}(\overline{\emptyset})$ for some m. Now since evaluation is continuous, for fixed m the function $f \mapsto f^{(m)}(\overline{\emptyset})$ is continuous as well. Hence by PFS, for our $b \in f^{(m)}(\overline{\emptyset})$ we find a $W \subseteq^{\text{fin}} f$ such that

$$b \in \overline{W}^{(m)}(\overline{\emptyset}) \subseteq Y(\overline{W}). \qquad \square$$

Let Y_ρ be the least fixed point operator associated with the information system $\mathbf{C}_\rho$, so $Y_\rho\colon (\mathbf{C}_\rho \to \mathbf{C}_\rho) \to \mathbf{C}_\rho$. We now show that Y_ρ is computable.

LEMMA. *Y_ρ is computable.*

PROOF. We need to show that Y_ρ is Σ^0_1-definable when viewed as a set of tokens. From the definitions of approximable maps, their correspondence with continuous functions, the definition of Y_ρ above, and the easily verified fact that $\overline{X}(\overline{Z}) = \overline{XZ}$, we obtain

$$\begin{aligned}
(X, a) \in Y_\rho \quad &\equiv \quad (X', a) \in Y_\rho \quad \text{for some } X' \subseteq^{\text{fin}} \overline{X} \\
&\equiv \quad a \in Y_\rho(\overline{X}) \\
&\equiv \quad \exists n (a \in \overline{X}^{(n)}(\overline{\emptyset})) \\
&\equiv \quad \exists n (a \in \overline{X^{(n)}\emptyset}) \\
&\equiv \quad \exists n (X^{(n)}\emptyset \vdash a).
\end{aligned}$$

Thus Y_ρ is Σ^0_1-definable if the relation $X^{(n)}\emptyset \vdash a$ is. But note that

$$X^{(n)}\emptyset \vdash a \equiv \exists \vec{Z}[\vec{Z} \text{ has length } n{+}1 \wedge Z_0 = \emptyset \wedge \forall i{<}n(Z_{i+1} = XZ_i) \wedge Z_n \vdash a].$$

Now the Z's are finite and encodable as integers using sequence-coding. Hence finite sequences of Z's are also encodable as integers. Furthermore the relation $Z' = XZ$ as defined in 4.4 is Σ^0_1-definable from a positive occurrence of the entailment relation $Z \vdash b$ at a type level lower than that of X. Therefore $X^{(n)}\emptyset \vdash a$ is Σ^0_1-definable from the entailment relation at this lower level, which in turn is then Σ^0_1-definable from an even lower level of entailment relation, etcetera. Therefore the relation $(X, a) \in \widehat{Y_\rho}$ can eventually be unravelled into an outright Σ^0_1 definition.

5.4. *Functionals recursive in* pcond *and* $\exists$. We first define some particular computable functionals. Recall that $\mathbb{N}$ is the flat information system whose tokens are the natural numbers, whose CON-sets are just the singletons $\{n\}$ and $\emptyset$, and whose entailment relation is simply $\ni$ (so $\{n\} \vdash m$ iff $n = m$, and $\emptyset \not\vdash m$). We shall often write $\perp$ for the empty set, and simply n for the singleton $\{n\}$ where no confusion should occur.

The *parallel conditional* pcond of type $\iota \to \iota \to \iota \to \iota$ is given by

$$\mathrm{pcond}(p, x, y) := \begin{cases} x & \text{if } p > 0 \\ y & \text{if } p = 0 \\ x & \text{if } x = y \\ \perp & \text{otherwise.} \end{cases}$$

From it we can define the *sequential conditional* cond of the same type by

$$\mathrm{cond}(p, x, y) := \mathrm{pcond}(p, \mathrm{pcond}(p, x, \perp), \mathrm{pcond}(p, \perp, y)).$$

Then

$$\mathrm{cond}(p, x, y) = \begin{cases} x & \text{if } p > 0 \\ y & \text{if } p = 0 \\ \perp & \text{if } p = \perp. \end{cases}$$

We can also define from pcond the *parallel or* of type $\iota \to \iota \to \iota$ given by

$$\vee(p, q) := \mathrm{pcond}(p, 1, q).$$

Then

$$\vee(p, q) = \begin{cases} 1 & \text{if } p > 0 \\ 1 & \text{if } q > 0 \\ 0 & \text{if } p = 0 \text{ and } q = 0 \\ \perp & \text{otherwise.} \end{cases}$$

Because these are at the lowest type level, it is necessary only for them to be monotone in order to be continuous, since PFS is trivial over the flat system $\mathbb{N}$. But their monotonicity is clear from their definitions. It is equally clear that each of these examples is computable. We only need to show that as relations (i.e. sets of tokens of the appropriate type) they are Σ^0_1-definable. But this is immediate: in the case of pcond the tokens are all 4-tuples (p, x, y, z) of numbers such that either $p > 0$ and $z = x$, or $p = 0$ and $z = y$, or $x = y = z$.

The *continuous existential quantifier* $\exists$ of type $(\iota \to \iota) \to \iota$ is defined as follows:

$$\exists(\varphi) := \begin{cases} 0 & \text{if } \varphi(\bot) = 0 \\ 1 & \text{if } \varphi(n) > 0 \text{ for some } n \\ \bot & \text{otherwise.} \end{cases}$$

Clearly $\exists$ is monotone. Furthermore, if $\exists(\varphi)$ has a defined value by either of the first two clauses, then it obviously has that same value on a finite subfunction of φ (only one bit of data is needed). Therefore $\exists$ satisfies PFS and hence is continuous.

Observe on the other hand, that the full existential quantifier, where the first line is replaced by '0 if $\varphi(n) = 0$ for every n', is *not* continuous. For if it were, then by PFS its value (namely 0) on the total constant-0 function would also be attained on some finite subfunction; this is impossible.

The continuous $\exists$-functional is computable since its tokens are all pairs (X, n) such that either $(\emptyset, 0) \in X$ and $n = 0$ or else there exists an m and a $k > 0$ such that $(\{m\}, k) \in X$ and $n = 1$. Clearly this is another Σ^0_1 condition.

DEFINITION. A partial continuous functional Φ of type $\rho_1 \to \ldots \to \rho_p \to \iota$ is said to be *recursive in* pcond *and* $\exists$ if it can be defined explicitly, for all arguments $\varphi_1, \ldots, \varphi_p$, by an equation

$$\Phi(\varphi_1, \ldots, \varphi_p) = M(\varphi_1, \ldots, \varphi_p)$$

where M is a λ-term built up from the variables $\varphi_1, \ldots, \varphi_p$ together with the constants 0, successor, predecessor, the fixed point operators Y_ρ, and pcond and $\exists$.

THEOREM. *(Plotkin). A partial continuous functional is computable if and only if it is recursive in* pcond *and* $\exists$.

PROOF The examples above together with the facts proved in 5.2 that

- (pure) λ-terms are computable and
- computable functionals are closed under application

show that every functional recursive in pcond and $\exists$ is computable. For the converse, we restrict attention to the case of a composed type $\rho_1 \to \ldots \to \rho_p \to \iota$; the details concerning product types are left as an exercise. Let Φ be computable of this type. Then Φ is a primitive recursively enumerable set of tokens

$$\Phi = \{\, (X^1_{f_1(n)}, \ldots, X^p_{f_p(n)}, g(n)) \mid n \in \mathbb{N} \,\}$$

where for each type ρ_j, $(X^j_i)_{i \in \mathbb{N}}$ is an enumeration of Con_{ρ_j}, and $f_1, \ldots, f_p$ and g are fixed primitive recursive functions. Henceforth we will drop the superscripts from the X's.

Let $\vec{\varphi} = \varphi_1, \ldots, \varphi_p$ be arbitrary continuous functionals of types $\rho_1, \ldots, \rho_p$ respectively. We show that Φ is definable by the equation

$$\Phi(\vec{\varphi}) = Y(w)(0)$$

with w of type $(\iota \to \iota) \to (\iota \to \iota)$ given by

$$w(\psi)(n) := \mathrm{pcond}(\mathrm{incons}(\varphi_1, f_1(n)) \vee \ldots \vee \mathrm{incons}(\varphi_p, f_p(n)), \psi(n+1), g(n)),$$

where the incons_{ρ_i}'s of type $\rho_i \to \iota \to \iota$ are the continuous functionals given by

$$\mathrm{incons}(\varphi, n) := \begin{cases} 1 & \text{if } \varphi \cup X_n \text{ is inconsistent} \\ 0 & \text{if } \varphi \supseteq X_n \\ \perp & \text{otherwise.} \end{cases}$$

We will prove in the lemma below that they are recursive in pcond and $\exists$; their definition will involve the functional $\exists$.

First suppose $\Phi(\vec{\varphi})$ is defined. By the definition of Φ this means that there is an n such that $\varphi_i \supseteq X_{f_i(n)}$ for every i. Let n_0 be the least such n. Then again from the definition of Φ, $\Phi(\vec{\varphi}) = g(n_0)$. We claim that

$$w^{k+1}(\lambda x \perp)(n_0 - k) = g(n_0) \quad \text{for every } k \leq n_0.$$

This is proved by induction on k. The base case $k = 0$ is immediate since $\varphi_i \supseteq X_{f_i(n_0)}$ hence $\mathrm{incons}(\varphi_i, f_i(n_0)) = 0$ for every i, so by definition of pcond, $w(\lambda x \perp)(n_0) = g(n_0)$. For the step case $k \to k+1$

$$\begin{aligned} w^{k+2}(\lambda x \perp)(n_0-k-1) &= w(w^{k+1}(\lambda x \perp))(n_0-k-1) \\ &= \mathrm{pcond}(\ldots, w^{k+1}(\lambda x \perp)(n_0-k), g(n_0-k-1)) \\ &= \mathrm{pcond}(\ldots, g(n_0), g(n_0-k-1)) \quad \text{by ind. hyp.} \\ &= g(n_0) \end{aligned}$$

with $\ldots = \mathrm{incons}(\varphi_1, f_1(n_0-k-1)) \vee \ldots \vee \mathrm{incons}(\varphi_p, f_p(n_0-k-1))$. The last step in this chain needs some explanation. Note that the first argument in pcond cannot have value 0, because this contradicts the minimality of n_0. There are two remaining cases: either the first argument of pcond is 1, in which case the value is the second argument, namely $g(n_0)$; or else the first argument of pcond is $\perp$, in which case $\varphi_i \cup X_{f_i(n_0-k-1)}$ is consistent for every i, and therefore

$$g(n_0) = \Phi(\overline{\varphi_i \cup X_{f_i(n_0-k-1)}}) = g(n_0-k-1).$$

Hence the value of the pcond-term is again $g(n_0)$, and this completes the proof of the claim.

From the claim for $k = n_0$ we obtain $w^{n_0+1}(\lambda x \perp)(0) = g(n_0)$, and since $Y(w)$ is the union of all the iterates $w^k(\lambda x \perp)$, it follows that $Y(w)(0) = g(n_0)$. We have therefore proved that if $\Phi(\vec{\varphi})$ is defined, then its value is $Y(w)(0)$. It remains to show the converse.

So suppose $Y(w)(0)$ is defined. Then there must be an n such that $\varphi_i \supseteq X_{f_i(n)}$ for every i. Otherwise we would have $w(\lambda x\bot) = \lambda x\bot$ by definition of w, since the first argument of pcond is never 0, so the value must be $\bot$; therefore $Y(w) = \lambda x\bot$, a contradiction. Let n_0 be the least such n. Then $\Phi(\vec{\varphi}) = g(n_0)$ by definition of Φ, and the above claim again applies to give $Y(w)(0) = g(n_0)$. Thus if $Y(w)(0)$ is defined then its value is $\Phi(\vec{\varphi})$. □

Note that from all fixed point operators Y_ρ we only made use of $Y_{\iota\to\iota}$, to construct the least fixed point of $w\colon(\iota\to\iota)\to(\iota\to\iota)$. The proof of the lemma below (which is all we need to finish the proof of Plotkin's theorem) wiil not make any essential use of Y (it only uses $Y_{\iota\to\iota}$ again to take care of primitive recursion).

Lemma. *The following functionals are recursive in* pcond *and* $\exists$*:*

1. en_ρ *of type* $\iota\to\iota\to\rho$ *such that* $\mathrm{en}(m)$ *enumerates all finitely generated extensions of* $\overline{X_m}$ *thus*

$$\begin{aligned} \mathrm{en}(m,\bot) &= \overline{X_m} \\ \mathrm{en}(m,n) &= \overline{X_n} \quad \textit{if } \overline{X_n} \supseteq \overline{X_m}. \end{aligned}$$

2. incons_ρ *of type* $\rho\to\iota\to\iota$*, defined by*

$$\mathrm{incons}(\varphi,n) := \begin{cases} 1 & \textit{if } \varphi\cup X_n \textit{ is inconsistent} \\ 0 & \textit{if } \varphi \supseteq X_n \\ \bot & \textit{otherwise.} \end{cases}$$

Proof. We show (1) and (2) by simultaneous induction on ρ. Again we restrict attention to the case of a composed type $\rho_1\to\ldots\to\rho_p\to\iota$; the details concerning product types are left as an exercise.

1. We first prove that en_ρ is recursive in pcond and $\exists$. For its definition we need to look in more detail into the definition of the sets X_m of type ρ.

For any type ρ, fix an enumeration $(X_n^\rho)_{n\in\mathbb{N}}$ of Con_ρ such that $X_0 = \emptyset$ and the following relations are primitive recursive:

$$\begin{aligned} &X_n \subseteq X_m, \\ &X_n \cup X_m \in \mathrm{Con}_\rho, \\ &X_n^{\rho\to\sigma} X_m^\rho = X_k^\sigma, \\ &X_n \cup X_m = X_k \qquad (\text{with } k = 0 \text{ if } X_n\cup X_m \notin \mathrm{Con}_\rho). \end{aligned}$$

We also assume an enumeration $(b_i^\rho)_{i\in\mathbb{N}}$ of the set of tokens of type ρ.

Note that any primitive recursive function f can be regarded as a continuous functional of type $\iota\to\ldots\to\iota\to\iota$ if we identify it with its strict extension. It is easy to see that any primitive recursive function can be represented in this way by a term of the simply typed λ-calculus involving 0, successor, predecessor, the least fixed point operators $Y_{\iota\to\iota}$ and the sequential conditional cond_ι. For instance, addition can be written as

$$m + n = Y_{\iota\to\iota}(\lambda\varphi\lambda x[\mathbf{if}\ x = 0\ \mathbf{then}\ m\ \mathbf{else}\ \varphi(x-1)+1\ \mathbf{fi}])n.$$

Let $\rho = \rho_1 \to \ldots \to \rho_p \to \iota$ and j, k and h be primitive recursive functions such that

$$X_m = \{\, (X_{j(m,1,l)}, \ldots, X_{j(m,p,l)}, k(m,l)) \mid l < h(m) \,\}.$$

en_ρ will be defined from an auxiliary functional Ψ of type $\rho_1 \to \ldots \to \rho_p \to \iota \to \iota \to \iota$ given by

$$\begin{aligned} \Psi(\vec{\varphi}, m, d, 0) &:= d \\ \Psi(\vec{\varphi}, m, d, l+1) &:= \mathrm{pcond}(p_l, \Psi(\vec{\varphi}, m, d, l), k(m,l)), \end{aligned}$$

where p_l denotes $\mathrm{incons}_{\rho_1}(\varphi_1, j(m,1,l)) \vee \ldots \vee \mathrm{incons}_{\rho_p}(\varphi_p, j(m,p,l))$. Hence

$$p_l = \begin{cases} 1 & \text{if } \varphi_i \cup X_{j(m,i,l)} \text{ is inconsistent for some } i = 1 \ldots p \\ 0 & \text{if } \varphi_i \supseteq X_{j(m,i,l)} \text{ for all } i = 1 \ldots p \\ \bot & \text{otherwise.} \end{cases}$$

Let

$$X_m^0 := \emptyset, \quad X_m^{l+1} := X_m^l \cup \{\, (X_{j(m,1,l)}, \ldots, X_{j(m,p,l)}, k(m,l)) \,\},$$

hence $X_m = X_m^{h(m)}$. We first show that

$$\Psi(\vec{\varphi}, m, \bot, l) = \overline{X_m^l}(\vec{\varphi}). \tag{$*$}$$

This is proved by induction on l. For $l = 0$ both sides $= \bot$. In the step $l \to l+1$ we distinguish three cases according to the possible values 1, 0 and $\bot$ of p_l. In case $p_l = 1$ by the definition of Ψ, the induction hypothesis and the fact that $p_l = 1$ implies that $\varphi_i \cup X_{j(m,i,l)}$ is inconsistent for some $i = 1 \ldots p$ we obtain

$$\Psi(\vec{\varphi}, m, \bot, l+1) = \Psi(\vec{\varphi}, m, \bot, l) = \overline{X_m^l}(\vec{\varphi}) = \overline{X_m^{l+1}}(\vec{\varphi}).$$

In case $p_l = 0$ by the definition of Ψ and the fact that $p_l = 0$ implies that $\varphi_i \supseteq X_{j(m,i,l)}$ for all $i = 1 \ldots p$ we obtain

$$\Psi(\vec{\varphi}, m, \bot, l+1) = k(m,l) = \overline{X_m^{l+1}}(\vec{\varphi}).$$

In case $p_l = \bot$ first assume that $\Psi(\vec{\varphi}, m, \bot, l) = \bot$. Then by induction hypothesis $\overline{X_m^l}(\vec{\varphi}) = \bot$, hence also $\overline{X_m^{l+1}}(\vec{\varphi}) = \bot$ (since we don't have $\varphi_i \supseteq X_{j(m,i,l)}$ for all $i = 1 \ldots p$), and $\Psi(\vec{\varphi}, m, \bot, l+1) = \bot$ by definition of pcond. If on the other hand $\Psi(\vec{\varphi}, m, \bot, l)$ is defined, then by induction hypothesis $\overline{X_m^l}(\vec{\varphi})$ is defined as well. Hence it must be $= k(m,l)$ (since $\varphi_i \cup X_{j(m,i,l)}$ is consistent for all $i = 1 \ldots p$, and X_m^{l+1} is consistent), hence by definition of pcond

$$\Psi(\vec{\varphi}, m, \bot, l+1) = k(m,l) = \overline{X_m^{l+1}}(\vec{\varphi}).$$

This completes the proof of $(*)$.

We now show

$$\Psi(\vec{\varphi}, m, d, l) = d, \tag{$**$}$$

provided d is the common value of all $k(m, l')$ such that $p_{l'} = 0$ or $= \bot$ (i.e. $\varphi_i \cup X_{j(m,i,l')}$ is consistent for all $i = 1 \dots p$). Again the proof is by induction on l. For $l = 0$ we have $\Psi(\vec{\varphi}, m, d, 0) = d$ by definition. In the step $l \to l+1$ we again distinguish cases according to the possible values of p_l. In case $p_l = 1$ we know that $\varphi_i \cup X_{j(m,i,l)}$ is inconsistent for some $i = 1 \dots p$, hence we have $\Psi(\vec{\varphi}, m, d, l+1) = \Psi(\vec{\varphi}, m, d, l) = d$ by induction hypothesis. In case $p_l = 0$ or $= \bot$ we know that that $\varphi_i \cup X_{j(m,i,l)}$ is consistent for all $i = 1 \dots p$. Then $k(m, l) = d$ by the choice of d, hence $\Psi(\vec{\varphi}, m, d, l+1) = d$ by induction hypothesis and the definition of pcond. This completes the proof of $(**)$.

We can now proceed with the proof of (1). Let

$$\begin{aligned} \Phi(\vec{\varphi}, m, d) &:= \Psi(\vec{\varphi}, m, d, h(m)) \\ \mathrm{en}(m, n, \vec{\varphi}) &:= \Phi(\vec{\varphi}, m, \Phi(\vec{\varphi}, n, \bot)) \end{aligned}$$

with the understanding that n is also allowed to take the value $\bot$. Note that $\Phi(\vec{\varphi}, n, \bot) = \overline{X_n}(\vec{\varphi})$ by $(*)$. The first property of en is now obvious, since

$$\mathrm{en}(m, \bot, \vec{\varphi}) = \Phi(\vec{\varphi}, m, \Phi(\vec{\varphi}, \bot, \bot)) = \Phi(\vec{\varphi}, m, \bot) = \overline{X_m}(\vec{\varphi}).$$

For the second property let $\overline{X_n} \supseteq \overline{X_m}$ and $\vec{\varphi}$ be given, and $d := \overline{X_n}(\vec{\varphi})$. Then by definition $\mathrm{en}(m, n, \vec{\varphi}) = \Phi(\vec{\varphi}, m, d)$, and $\Phi(\vec{\varphi}, m, d) = d$ follows from $(**)$.

2. Let $\rho = \sigma \to \tau$ and fix primitive recursive functions f and g such that the i-th token at type ρ is $b_i^\rho = (X_{f(i)}^\sigma, b_{g(i)}^\tau)$. We will define the functionals incons_ρ from similar functionals ic_ρ of type $\rho \to \iota \to \iota$ given by

$$\mathrm{ic}(\varphi, i) := \begin{cases} 1 & \text{if } \varphi \cup \{b_i\} \text{ is inconsistent} \\ 0 & \text{if } \varphi \supseteq \{b_i\} \\ \bot & \text{otherwise.} \end{cases}$$

In order to show that the ic's are recursive in pcond and $\exists$ we carry out the following computations.

$$\begin{aligned} \mathrm{ic}_\rho(\varphi, i) = 1 &\equiv \varphi \cup \{b_i\} \text{ is inconsistent} \\ &\equiv \varphi \cup \{(X_{f(i)}, b_{g(i)})\} \text{ is inconsistent} \\ &\equiv \exists n(\overline{X_n} \supseteq X_{f(i)} \text{ and } \varphi(\overline{X_n}) \cup \{b_{g(i)}\} \text{ is inconsistent}) \\ &\equiv \exists n(\varphi(\mathrm{en}_\sigma(f(i), n)) \cup \{b_{g(i)}\} \text{ is inconsistent}) \\ &\equiv \exists n(\mathrm{ic}_\tau(\varphi(\mathrm{en}_\sigma(f(i), n)), g(i)) = 1) \end{aligned}$$

and

$$\begin{aligned} \mathrm{ic}_\rho(\varphi, i) = 0 &\equiv b_i \in \varphi \\ &\equiv (X_{f(i)}, b_{g(i)}) \in \varphi \\ &\equiv b_{g(i)} \in \varphi(\overline{X_{f(i)}}) \\ &\equiv \mathrm{ic}_\tau(\varphi(\mathrm{en}_\sigma(f(i), \bot)), g(i)) = 0. \end{aligned}$$

Hence we can define

$$\mathrm{ic}_\rho(\varphi, i) := \exists(\lambda n.\mathrm{ic}_\tau[\varphi(\mathrm{en}_\sigma(f(i), n)), g(i)]).$$

We still have to define incons_ρ from ic_ρ. Let

$$\begin{aligned} \mathrm{ic}^*(\varphi, n, 0) &:= 0 \\ \mathrm{ic}^*(\varphi, n, l+1) &:= \mathrm{ic}^*(\varphi, n, l) \vee \mathrm{ic}(\varphi, j(n, l)), \end{aligned}$$

where $j(n, l)$ is defined by $X_n^\rho = \{\, b_{j(n,l)} \mid l < h(n)\,\}$. It is now easy to see that

$$\mathrm{incons}_\rho(\varphi, n) = \mathrm{ic}_\rho^*(\varphi, n, h(n)).$$

Note that one needs the coherence of $\mathrm{C{\scriptsize ON}}_\rho$ here (cf. corollary 4.10). Note also that we do need the parallel or in the definition of ic^*. □

6 Bounded fixed point operators

Having seen that there is a quite nice internal definition of the computable functionals over the partial continuous functionals, we now come back to our subject of classifying recursive functions. The idea is simply to combine the power and elegance of treating recursion by means of the fixed point operator with the advantages of having a higher order term language with more 'controlled' forms of higher type recursion, like Gödel's system T of [20] for the primitive recursive functionals. Moreover, we want to have conversion rules that actually carry out the computations, and are terminating and confluent. To achieve this goal, we introduce bounded versions of the fixed point operators, depending on a given well-ordering. We essentially follow [58] here, but – in order to simplify our arguments in section 7 – we do not allow recursion w.r.t. a measure function.

6.1. Let a well-ordering $\prec$ of the natural numbers be given. We want the bounded fixed point operator $[\![\mathrm{Y}_{\rho,\prec}]\!]$ to satisfy, for $x \neq \emptyset$,

$$[\![\mathrm{Y}_{\rho,\prec}]\!](G, x) := G\left(\lambda y, \vec{y}.\begin{cases} [\![\mathrm{Y}_{\rho,\prec}]\!](G, y, \vec{y}) & \text{if } y \prec x \\ \emptyset & \text{otherwise} \end{cases}, x\right).$$

This equality, when viewed as a conversion rule from left to right, clearly gives rise to an infinite reduction sequence: just expand the subterm corresponding to $[\![\mathrm{Y}_{\rho,\prec}]\!](G, y)$ by the same rule again, and carry on doing that. Hence in order to achieve termination we have to make use of the fact that $[\![\mathrm{Y}_{\rho,\prec}]\!](G, y)$ occurs in a 'guarded' context only. Therefore we restrict our conversion rules and allow conversion of a term $\mathrm{cond}MNL$ only if M is a constant (i.e. a numeral $\mathbf{n}$ or $\perp$). Then clearly the infinite reduction sequence above can not be formed any more, but still we can expect that any closed term of ground type reduces to a numeral.

It is shown below that every closed term of ground type involving only bounded fixed point operators is strongly normalizable. Hence the object 'undefined' in the model can be viewed as a finite error. This is in contrast to the term language PCF of [46], where the value undefined is to be interpreted as a nonterminating computation. The addition of constants for finite errors to a term language has also been considered by Dosch in [11]; however, Dosch only deals with strict functionals.

6.2. Let us now carry out this program. We fix a well-ordering $\prec$ of the natural numbers. First we have to extend the term language by

- the sequential conditional cond of type $\iota \to \iota \to \iota \to \iota$,
- $c_\prec$ and $c_=$ of type $\iota \to \iota \to \iota$,
- for each type ρ, the constant $\mathrm{Y}_{\rho,\prec}$ of type $[(\iota \to \rho) \to \iota \to \rho] \to \iota \to \rho$.

Let us call the resulting term language $\mathrm{PCF}_\prec$.

Recall that the sequential conditional cond was definable in our original language. However, we must introduce cond as a separate constant now, since it will be given a special status in our operational semantics, which makes it possible to use it as a 'guard' blocking infinite reduction sequences.

NOTATION. In this section we have been (and will continue to be) slightly sloppy in not distinguishing notationally a constant of the term language $\mathrm{PCF}_\prec$ (e.g. cond) from the partial continuous functional it denotes (e.g. $[\![\mathrm{cond}]\!]$). However, it should always be clear from the context what is meant. Recall also that we normally write $\perp$ for the empty set, and simply n for the singleton $\{n\}$ where no confusion should occur.

For any of the new constants $c_\prec, c_=, \mathrm{Y}_{\rho,\prec}$, we define its value as follows.

$$c_\prec(x,y) := \begin{cases} 1 & \text{if } x \prec y \text{ (and } x, y \text{ are both defined)} \\ 0 & \text{if } x \not\prec y \text{ (and } x, y \text{ are both defined)} \\ 0 & \text{if either } x \text{ of } y \text{ is undefined (i.e. } = \perp), \end{cases}$$

and similarly for $c_=$.

$\mathrm{Y}_{\rho,\prec}$ is to be an object of type $[(\iota \to \rho) \to \iota \to \rho] \to \iota \to \rho$. So let $\rho = \rho_1 \to \ldots \to \rho_n \to \iota$ and objects G of type $(\iota \to \rho) \to \iota \to \rho$, x of type ι and $\vec{x}$ of type $\vec{\rho}$ be given. We define $\mathrm{Y}_{\rho,\prec}(G, x, \vec{x})$ by (transfinite) $\prec$-recursion on x, as follows. If x is undefined, then $\mathrm{Y}_{\rho,\prec}(G, x, \vec{x})$ is undefined. Otherwise, assume that for all y with $y \prec x$ and all $\vec{y}$ the value $\mathrm{Y}_{\rho,\prec}(G, y, \vec{y})$ is already known. Then let

$$\mathrm{Y}_{\rho,\prec}(G, x, \vec{x}) := G([\mathrm{Y}_{\rho,\prec}(G)]_{\prec x}, x, \vec{x}),$$

where for any F of type $\iota \to \rho$ the restriction $[F]_{\prec x}$ of F below x is defined by

$$[F]_{\prec x}(y, \vec{y}) := \begin{cases} F(y, \vec{y}) & \text{if } y \prec x \\ \emptyset & \text{otherwise.} \end{cases}$$

6.3. *Operational semantics.* The operational semantics is given by the following conversion rules. Recall that $\mathbf{m}$ is the m-th numeral, so $[\![\mathbf{m}]\!] = m$.

(i) $(\lambda x^\rho M)N \to M[N/x]$.

(ii) $\dfrac{M \to M'}{MN \to M'N}$.

(iii) $\mathrm{Y}_{\rho,\prec} NK\vec{N} \to \mathrm{cond}(c_= KK)[N(\lambda y, \vec{y}.\mathrm{cond}(c_\prec yK)(\mathrm{Y}_{\rho,\prec} Ny\vec{y})\bot)K\vec{N}]\bot$.

(iv) $\mathrm{cond}\mathbf{k}MN \to M$ if $k > 0$; $\mathrm{cond}0MN \to N$; $\mathrm{cond}\bot MN \to \bot$.

(v) $\dfrac{M \to M'}{\mathrm{cond}M \to \mathrm{cond}M'}$.

(vi) $\mathrm{pcond}\mathbf{k}MN \to M$ if $k > 0$; $\mathrm{pcond}0MN \to N$; $\mathrm{pcond}MNN \to N$; $\mathrm{pcond}\bot M\bot \to \bot$; $\mathrm{pcond}\bot\bot N \to \bot$; $\mathrm{pcond}\bot cd \to \bot$ if c and d are different constants or numerals.

(vii) $\dfrac{M \to M'}{\mathrm{pcond}M \to \mathrm{pcond}M'}$; $\dfrac{N \to N'}{\mathrm{pcond}MN \to \mathrm{pcond}MN'}$; $\dfrac{L \to L'}{\mathrm{pcond}MNL \to \mathrm{pcond}MNL'}$.

(viii) $c_\prec \mathbf{mn} \to \begin{cases} 1 & \text{if } m \prec n \\ 0 & \text{otherwise} \end{cases}$; $c_\prec \bot N \to \bot$; $c_\prec M\bot \to \bot$; similarly for $c_=$. $\mathrm{S}\bot \to \bot$.

(ix) $\dfrac{M \to M'}{c_\prec M \to c_\prec M'}$; $\dfrac{N \to N'}{c_\prec MN \to c_\prec MN'}$; similarly for $c_=$ and S.

It might seem that the simpler rule

(iii)' $\mathrm{Y}_{\rho,\prec} NK\vec{N} \to N(\lambda y, \vec{y}.\mathrm{cond}(c_\prec yK)(\mathrm{Y}_{\rho,\prec} Ny\vec{y})\bot)K\vec{N}$

for the $\mathrm{Y}_{\rho,\prec}$-operator would suffice. However, it does not in case K evaluates to $\bot$. For then the outcome of the simplified rule (iii') depends on the form of N and might be defined, which would be inconsistent with our denotational semantics.

Note that we do *not* have

- $\dfrac{N \to N'}{MN \to MN'}$
- $\dfrac{M \to M'}{\lambda xM \to \lambda xM'}$,

for then following an application of rule (iii) for the bounded fixed point operator we could easily construct an infinite reduction sequence.

6.4. *Normalization via computability predicates.* We now prove that every closed term M of ground type can be computed by the rules of 6.3, i.e. that any reduction sequence starting with M terminates. It clearly follows from the form of our conversion rules that the final term in any reduction sequence must be a constant or a numeral. Note that it is only in the range of the constants other than the sequential conditional cond that we have a choice of where to reduce.

For the proof we use W.W. Tait's method of computability predicates. So for every type ρ we define what it means for a term of type ρ to be computable. The definition is by induction on types.

1. A closed term M of ground type is computable if any reduction sequence starting with M terminates.

2. A closed term M of type $\rho \to \sigma$ is computable if for every closed term N of type ρ, if N is computable, then so is MN.

3. A term M containing at least one free variable is computable if every type respecting substitution of closed computable terms for the free variables in M yields a computable term.

THEOREM. *Every term M is computable.*

PROOF. By induction on M.

Case x. Trivial.

Case MN. Let $\vec{L}$ be a sequence of closed computable terms. We have to show that $M[\vec{L}]N[\vec{L}]$ is computable. But this holds, since by induction hypothesis we know that $M[\vec{L}]$ as well as $N[\vec{L}]$ are computable.

Case $\lambda x M$. Let $\vec{L}$ be a sequence of closed computable terms. We have to show that $\lambda x M[\vec{L}]$ is computable. So let $N, \vec{M}$ be closed computable terms. We must show that $(\lambda x M[\vec{L}])N\vec{M}$ is computable, i.e. that every reduction sequence starting with that term terminates. But from the rules of our operational semantics it follows that the second member of any such sequence must be $M[N, \vec{L}]\vec{M}$, which is computable by induction hypothesis.

Case $\mathrm{Y}_{\rho,\prec}$. Let $N, K, \vec{N}$ be closed computable terms. We have to show that $\mathrm{Y}_{\rho,\prec}NK\vec{N}$ is computable, i.e. that every reduction sequence starting with that term terminates. Since K is computable and of type ι, we know that K reduces to a constant or a numeral c; we use an auxiliary (transfinite) $\prec$-induction on (the value of) c. If $c = \bot$, then every reduction sequence starting with $\mathrm{Y}_{\rho,\prec}NK\vec{N}$ must, after its first two steps, work on reducing $c_{=}KK$ until this term has been reduced to a constant or a numeral; this follows from the form of the conversion rules for the sequential conditional. Now since by assumption K reduces to $\bot$ we must get $\bot$, and hence $\mathrm{Y}_{\rho,\prec}NK\vec{N}$ reduces to $\bot$. So assume $c = \mathbf{n}$. Then as before we can see that every reduction sequence starting with $\mathrm{Y}_{\rho,\prec}NK\vec{N}$ must come across

$$N(\lambda y, \vec{y}.\mathrm{cond}(c_{\prec}yK)(\mathrm{Y}_{\rho,\prec}Ny\vec{y})\bot)K\vec{N}.$$

Since N is computable by assumption, it suffices to show that the closed term

$$\lambda y, \vec{y}.\mathrm{cond}(c_{\prec} yK)(\mathrm{Y}_{\rho,\prec} Ny\vec{y})\bot$$

is computable. So let $L, \vec{L}$ be a closed computable term. We must show that

$$\mathrm{cond}(c_{\prec} LK)(\mathrm{Y}_{\rho,\prec} NL\vec{L})\bot$$

is computable. Since L is computable by assumption, we know that L reduces to a constant or a numeral d. If $d = \bot$, then $c_{\prec}LK$ must reduce to $\bot$ and hence the whole term must reduce to $\bot$. So assume $d = \mathbf{m}$. Then the whole term must reduce to

$$\mathrm{cond}(c_{\prec}\mathbf{mn})(\mathrm{Y}_{\rho,\prec} NL\vec{L})\bot$$

If $m \prec n$ holds, then $c_{\prec}\mathbf{mn} \to \mathbf{k}$ for some $k > 0$ and hence the whole term must reduce to $\mathrm{Y}_{\rho,\prec} NL\vec{L}$, which by hypothesis of our auxiliary $\prec$-induction is computable. If $m \prec n$ does not hold, then $c_{\prec}\mathbf{mn} \to 0$ and hence the whole term reduces to $\bot$.

Case cond, pcond. Let M, N, L be closed computable terms. We have to show that $\mathrm{cond}MNL$ and $\mathrm{pcond}MNL$ are computable, i.e. that every reduction sequence starting with either of them terminates. But this clearly holds, since otherwise one of the terms M, N, L would have an infinite reduction sequence.

For the other constants of $\mathrm{P{\scriptsize CF}}_{\prec}$ the argument is similar. □

Note that this proof involves computability predicates of arbitrary types and hence is not formalizable in Peano arithmetic in all finite types. It is well-known that this is necessarily so for every proof of a result implying the existence of a normal form for every term M.

6.5. *Reduction of* $\mathrm{Y}_{\rho,<}$ *to higher order primitive recursion.* We first consider the case where the well-ordering $\prec$ is the standard ordering $<$ of the natural numbers. Then the bounded fixed point operator is definable by means of primitive recursion, as follows.

Define a functional H_ρ of type $[(\iota \to \rho) \to \iota \to \rho] \to \iota \to \iota \to \rho$ by primitive recursion, as in Gödel's system T [20]:

$$\begin{aligned}
H_\rho(G, \bot) &:= \lambda y, \vec{y}\,\bot,\\
H_\rho(G, 0) &:= \lambda y, \vec{y}\,\bot,\\
H_\rho(G, n+1) &:= K_\rho(G, n, H_\rho(G, n)),
\end{aligned}$$

where K_ρ of type $[(\iota \to \rho) \to \iota \to \rho] \to \iota \to (\iota \to \rho) \to \iota \to \rho$ is defined by

$$K_\rho(G, x, z, y, \vec{y}) := \mathrm{cond}(c_{=}(y, x), G(z, x, \vec{y}), \mathrm{cond}(c_{<}(y, x), z(y, \vec{y}), \bot)).$$

Then we have, by an easy induction on n,

LEMMA.
$$H_\rho(G,n,m) = \begin{cases} \mathrm{Y}_{\rho,<}(G,m) & \text{if } m < n \\ \lambda\vec{y}\,\bot & \text{otherwise.} \end{cases}$$

PROOF. It clearly suffices to prove this for $n+1$. In case $m = n$ we have

$$\begin{aligned} H(G,n+1,n,\vec{y}) &= K(G,n,H(G,n),n,\vec{y}) \\ &= G(H(G,n),n,\vec{y}) \\ &= G([\mathrm{Y}_{\rho,<}(G)]_{<n},n,\vec{y}) \quad \text{by ind. hyp. for } n \\ &= \mathrm{Y}_{\rho,<}(G,n,\vec{y}), \end{aligned}$$

and in case $m < n$

$$\begin{aligned} H(G,n+1,m,\vec{y}) &= K(G,n,H(G,n),m,\vec{y}) \\ &= H(G,n,m,\vec{y}) \\ &= \mathrm{Y}_{\rho,<}(G,m,\vec{y}) \quad \text{by ind. hyp. for } n. \end{aligned}$$

In case $n < m$ both sides are undefined. □

Hence $\mathrm{Y}_{\rho,<}$ is explicitly definable from H_ρ, by

$$\mathrm{Y}_{\rho,<}(G,x) = H_\rho(G,\mathrm{S}(x),x).$$

6.6. *Reduction of higher order primitive recursion to* $\mathrm{Y}_{\rho,\prec}$. Let us now consider a general well-ordering $\prec$. We show that, under some mild assumptions on $\prec$, higher order primitive recursion can be reduced to the bounded fixed point operator $\mathrm{Y}_{\rho,\prec}$. The assumptions on $\prec$ say that $<$ can be embedded by means of a primitive recursive embedding-projection pair into $\prec$, i.e. that there are primitive recursive functions h, h' such that

$$x < y \to h(x) \prec h(y) \tag{$\prec$1}$$
$$h'(h(x)) = x. \tag{$\prec$2}$$

From now on let us assume that all well-orderings $\prec$ considered satisfy ($\prec$1) and ($\prec$2).

LEMMA. *(Reduction of higher order primitive recursion to* $\mathrm{Y}_{\rho,\prec}$*). Let G of type ρ and H of type $\rho \to \iota \to \rho$ be given. Define*

$$\begin{aligned} G_1(u,y) &:= \begin{cases} G & \text{if } y = h(0) \\ H(u(h(x)),x) \text{ with } x = h'(y) - 1, & \text{otherwise,} \end{cases} \\ F_1 &:= \mathrm{Y}_{\rho,\prec}(G_1), \\ F(x) &:= F_1(h(x)). \end{aligned}$$

Then F satisfies the primitive recursion equations w.r.t. G and H, i.e.

$$\begin{aligned} F(\bot) &= \lambda\vec{y}\,\bot, \\ F(0) &= G, \\ F(n+1) &= H(F(n), n). \end{aligned}$$

PROOF. $F(0) = F_1(h(0)) = G_1([F_1]_{\prec h(0)}, h(0)) = G$, and

$$\begin{aligned} F(n+1) &= F_1(h(n+1)) \\ &= G_1([F_1]_{\prec h(n+1)}, h(n+1)) \\ &= H([F_1]_{\prec h(n+1)}(h(n)), n) \\ &= H(F_1(h(n)), n) \\ &= H(F(n), n). \end{aligned}$$ □

6.7. *Constants introduced by bounded recursion.* In section 7 it will be more convenient to work with an equivalent version of the bounded fixed point operators, namely with functionals introduced by bounded recursion. By this we mean a definition of a functional F from a given functional G, of the form

$$F(x) := \begin{cases} \lambda\vec{y}\,\bot & \text{if } x = \bot \\ G([F]_{\prec x}, x) & \text{otherwise,} \end{cases}$$

with $[F]_{\prec x}$ as in 6.2. Then clearly $F = \mathrm{Y}_{\rho,\prec}(G)$; hence the fixed point operators are strong enough to allow this form of $\prec$-recursive definition. In fact, a fragment of the term language $\mathrm{P{\scriptstyle CF}}_\prec$ suffices, where $\mathrm{Y}_{\rho,\prec}$ is only allowed to occur in a context $\mathrm{Y}_{\rho,\prec}N$ with N a *closed* term.

We now prove that this restriction of the use of fixed point operators is not essential.

LEMMA. *$\mathrm{Y}_{\rho,\prec}$ is definable by explicit definitions and one $\prec$-recursion yielding a functional of the same type level as $\mathrm{Y}_{\rho,\prec}$.*

PROOF. Let

$$\begin{aligned} G_0(u, x, G) &:= G(\lambda y.u(y, G), x), \\ F(x) &:= \begin{cases} \lambda\vec{y}\,\bot & \text{if } x = \bot \\ G_0([F]_{\prec x}, x) & \text{otherwise,} \end{cases} \\ H(G, x) &:= F(x, G). \end{aligned}$$

Then $H(G, \bot) = F(\bot, G) = \lambda\vec{y}\,\bot$ and for $x \neq \bot$

$$\begin{aligned} H(G, x) &= G_0([F]_{\prec x}, x, G) \\ &= G(\lambda y.[F]_{\prec x}(y, G), x) \\ &= G\left(\lambda y \begin{cases} F(y, G) & \text{if } y \prec x \\ \lambda\vec{y}\,\bot & \text{otherwise} \end{cases}, x\right) \end{aligned}$$

$$= G\left(\lambda y \begin{cases} H(G,y) & \text{if } y \prec x \\ \lambda \vec{y}\,\bot & \text{otherwise} \end{cases}, x\right)$$
$$= G([H(G)]_{\prec x}, x).$$

Hence $H = \mathrm{Y}_{\rho,\prec}$. □

Note that a $\prec$-recursion of the form $F(x) = G([F]_{\prec x}, x)$ might be a 'nested' (see Tait [64]) recursion; e.g. $G(f,x) = f(f(x))$ is allowed.

7 Elimination of detours through higher types by transfinite recursion

We have seen in 5.1 that a detour through higher types (specifically the integer types ρ_n) makes it possible to give a rather perspicious definition of the generating functions F_α, $\alpha < \varepsilon_0$, of the extended Grzegorczyk hierarchy: every F_α can be built from iteration functionals by application alone, in a way determined by the Cantor normal form of α to base ω. Now our goal is to establish some kind of converse to this result, namely that detours through higher types can be eliminated at the expense of raising the ordinals of the transfinite recursions involved. The first result of this kind is due to Kreisel [29]; here we discuss a generalization that first appeared in [55, 56]. We will also sketch a new proof of this result, by adapting a method of Buchholz [5].

7.1. In order to formulate the result we first need some preparations. In what follows it will be cumbersome to construct explicitly the well-orderings to which our norms need to refer. But since we have to work with 'canonical' well-orderings of concrete order types like ε_0, Γ_0 or the Bachmann-Howard ordinal anyway (cf. the introduction to section 3), and since such ordinals can be coded by natural numbers in a straightforward way, we may think as well of norms as mappings directly into the ordinals, e.g. those $<\varepsilon_0$. This will be done in the rest of the paper.

Similarly, instead of $\mathrm{Y}_{\rho,\prec}$, $\prec$-recursion and $\mathrm{PCF}_\prec$ we will speak of $\mathrm{Y}_{\rho,\alpha}$, α-recursion and PCF_α, where α (e.g. $<\varepsilon_0$) is thought of as representing a standard well-ordering of this order type.

7.2. THEOREM. *(Trade-off theorem). Let M be a closed term of PCF_α of type level $n+1$ such that every $\mathrm{Y}_{\rho,\alpha}$ appearing in M has $\mathrm{lev}(\rho) \le n+m+1$ with $m \ge 1$. Then one can find $\beta < 2_{m+1}(\alpha+\omega)$ and a closed term M' of PCF_β involving just one $\mathrm{Y}_{\sigma,\beta}$ such that $[\![M]\!] = [\![M']\!]$ and $\mathrm{lev}(\sigma) \le n+1$.*

Here $2_m(\gamma)$ is the m-fold iteration of the ordinal function $2^{\cdot}$ to γ, i.e. $2_0(\gamma) := \gamma$, $2_{m+1}(\gamma) := 2^{2_m(\gamma)}$.

7.3. We now sketch the proof given in [55, 56]. First, Tait has observed in [65] that every α-recursive functional or – as we should say here – every term M in PCF_α can be represented by an infinite term t_M, built up from typed variables and constants by application, λ-abstraction and the formation of sequences $\langle t_i\rangle_{i\in\mathbb{N}}$ of type $\iota\to\iota$ with terms t_i of type ι.

For instance, if F is defined by α-recursion at type ρ, i.e. $F = Y_{\rho,\alpha}N$ with N a closed term of type $(\iota\to\rho)\to\iota\to\rho$, then one can represent $F\zeta$ by

$$t_\zeta := N\langle t_{\zeta\eta}\rangle_\eta \quad \text{with } t_{\zeta\eta} := \begin{cases} t_\eta & \text{if } \eta<\zeta \\ \lambda\vec{x}\,\bot & \text{otherwise.}\end{cases}$$

So F can be represented by $t_F := \langle t_\zeta\rangle_\zeta$; here a sequence $\langle s_i\rangle_{i\in\mathbb{N}}$ of terms s_i of a non ground type is an abbreviation for $\lambda x\vec{x}\,\langle s_i\vec{x}\rangle_{i\in\mathbb{N}}x$.

Now let M be a closed term of PCF_α of type level $n+1$ such that every $\mathrm{Y}_{\rho,\alpha}$ appearing in M has level $\leq n+m+1$ with $m\geq 1$. We may assume that M is β-normal, and moreover (by 6.7) that every $Y_{\rho,\alpha}$ in M appears in a context $Y_{\rho,\alpha}N$ with N closed. Now expand every subterm $Y_{\rho,\alpha}N$ as explained above. One can see easily that the resulting infinite term t_M has depth $|t_M|<\alpha+\omega$ and rank $\mathrm{Rk}(t_M)\leq n+m+2$; here the rank $\mathrm{Rk}(t)$ of an infinite term t is defined to be the supremum of the type levels of all subterms λxs in a context $(\lambda xs)r$. One can define a reduction relation (essentially as in Tait [65], but using β-conversions only) such that every infinite term t of $\mathrm{Rk}(t)\leq k+1$ can be reduced to a t' with $\mathrm{Rk}(t')\leq k$ and $|t'|\leq 2^{|t|}$. Hence, the above M of type level $n+1$ can be represented by an infinite term t_M^* with $\mathrm{Rk}(t_M^*)\leq n+1$ and depth $|t_M^*|<2_{m+1}(\alpha+\omega)$.

We consider now finite numbers or codes for infinite terms, and define valuation terms $\mathrm{VAL}_\rho^{\alpha,S}$ with the following property. Let $\ulcorner t\urcorner$ be a code of a closed infinite term t of type ρ and depth $|t|<\alpha$, all of whose subterms have types from a finite set S. Then $\mathrm{VAL}_\rho^{\alpha,S}\ulcorner t\urcorner$ represents the same functional as t. The definition of the $\mathrm{VAL}_\rho^{\alpha,S}$, $\rho\in S$, is by simultaneous α-recursion.

Now let M be as before. We first obtain a number $\ulcorner t_M\urcorner$ of an infinite term representing M, such that $\mathrm{VAL}_\rho^{\alpha+k,S}\ulcorner t_M\urcorner$ for suitable k,S represents the same functional as M. Corresponding to the reduction of the term t_M of $\mathrm{Rk}(t_M)\leq n+m+2$ and with depth $|t_M|<\alpha+\omega$ to a term t_M^* with $\mathrm{Rk}(t_M^*)\leq n+1$ and with depth $|t_M^*|<2_{m+1}(\alpha+\omega)$ we construct a function RED^* such that

$$\mathrm{VAL}_\rho^{\alpha+k,S}\ulcorner t_M\urcorner = \mathrm{VAL}_\rho^{\beta,S_{n+1}}(\mathrm{RED}^*\ulcorner t_M\urcorner)$$

with $\beta<2_{m+1}(\alpha+\omega)$ and a finite set S_{n+1} of types with levels $\leq n+1$. The function RED^* turns out to be primitive recursive; a similar situation occurs in the theory of Kleene's $\mathcal{O}$, where $+_{\mathcal{O}}$ can be chosen primitive recursive; cf. [25]. Since $\mathrm{VAL}_\rho^{\beta,S_{n+1}}$ is obtained using β-recursion but without auxiliary functionals of type level $>n+1$ we have the desired result.

7.4. In the rest of the paper we sketch a different proof of the trade-off theorem, by adapting a method of Buchholz [5] originally developed for well-founded sequent style derivations in ω-arithmetic. In that paper, the basic idea is to introduce a primitive recursive notation system for well-founded ω-derivations in the same way as one usually introduces an ordinal notation system as a system of terms generated from constants for particular ordinals by function symbols corresponding to certain ordinal functions. A less-than relation between ordinal notations is then derived from the properties of the denoted ordinals and ordinal functions. Instead of ordinals Buchholz considers well-founded derivations in the standard system $\mathbf{Z}^\infty$ of ω-arithmetic, with an unrestricted ω-rule. Each derivation in Peano-arithmetic $\mathbf{Z}$ can be viewed as a notation for a particular $\mathbf{Z}^\infty$-derivation; so $\mathbf{Z}$-derivations can play the role of constants here. The role of ordinal functions is taken over by the standard operators for cut elimination in infinite derivations, as introduced in Schütte [51], Tait [66] and Mints [36, 37]. Buchholz then introduces a system $\mathbf{Z}^*$ of *finite* terms built from derivations in $\mathbf{Z}$ by these function symbols. Each term (or notation) $h \in \mathbf{Z}^*$ then clearly denotes an infinite derivation $[\![h]\!] \in \mathbf{Z}^\infty$.

The main technical achievement of Buchholz' paper is that without explicit use of indices for (primitive) recursive functions in a coding of infinite derivations he obtains rather elegantly defined *primitive* recursive functions $\cdot[\cdot]$ and $o(\cdot)$ such that $h[n]$ denotes the n-th premise of the infinite derivation $[\![h]\!]$ denoted by h, and $o(h)$ gives an ordinal bound for $[\![h]\!]$. It is quite clear that one can give *recursive* definitions of these functions, since the corresponding operators on infinite derivations in $\mathbf{Z}^\infty$ are defined by transfinite recursion. The trick to obtain primitive recursive such functions is the usual one, namely to introduce some delay operators, in the present case the repetition rule introduced by Mints [36].

Here we transfer Buchholz' approach to the somewhat more flexible setting of infinite (typed) terms. We then have to deal with β-contraction and hence with substitutions, and this will cause some additional complication. However, one benefit is that this approach allows applications in recursion theory as well as in proof theory, since by the well-known Curry-Howard correspondence every natural (infinite) deduction can be viewed as an (infinite) term with formulas as types. In the present paper we restrict ourselves to just one application of the method in recursion theory, namely the trade-off theorem mentioned above. More details including another application can be found in [2].

7.5. *Infinite terms with the repetition rule.* Infinite terms are defined inductively, as in Tait [65]. They are built from

- typed variables x^ρ by
- constant application $c\vec{t}$,
- λ-abstraction $\lambda x^\rho t$,
- application ts,
- sequence formation $\langle t_i^\iota \rangle_{i\in\mathbb{N}}$, and
- repetition $\textsc{Rep}(t)$.

$\mathrm{Tp}(t)$ denotes the type of t. We assume that finitely many constants c denoting primitive recursive functions are given, where each such function expects $n \geq 0$

arguments of ground types $\iota_1, \ldots, \iota_n$ and yields a value of ground type ι; in addition, for every $k \in \mathbb{N}^{\perp}$ we have a constant $\mathbf{k}$. The *level* of a term is the level of its type.

The *rank* of an infinite term is defined to be the supremum of the levels of all subterms of the form $\lambda x t$ or $\textsc{Rep}(t)$ in a context $(\lambda x t)s$ or $(\textsc{Rep}(t))s$. Substitution, β-conversion and a function $\textsc{Red}(t)$ reducing the rank of t by 1 are defined as usual. However, at certain points of the inductive definition one has to insert applications of the repetition rule; this is needed to allow later a primitive recursive recognition which rule had been applied at a given node. It turns out that the depth $|\textsc{Red}(t)|$ can only be estimated by $3^{|t|}$ (not $2^{|t|}$); this leads to slightly weaker bounds in the trade-off theorem.

7.6. *Notation systems for infinite terms.* We now introduce notation systems for infinite terms. From each notation a of a term t we want to be able to read off the type of t (as $\mathrm{tp}(a)$), the rule applied last to form t (as $\mathrm{rl}(a)$) and for each $n < \mathrm{arity}(\mathrm{rl}(a))$ the n-th predecessor of t (as $a[n]$). We will also introduce a bound function fv, which for every notation a of a term t gives an upper bound $\mathrm{fv}(a)$ on the variables free in t. This information will be needed when we define substitution, for then a certain renaming of bound variables will be necessary.

A *notation system* $\mathcal{T} = (T, \mathrm{tp}, \mathrm{fv}, \mathrm{rl}, \cdot[\cdot])$ is given by a primitive recursive non-empty set $T \subseteq \mathbb{N}$ together with four functions

$$\begin{aligned}
&\mathrm{tp}\colon T \to \text{(codes of) types},\\
&\mathrm{fv}\colon T \to \text{(codes of) finite sets of variables},\\
&\mathrm{rl}\colon T \to \text{(codes of) rules},\\
&\cdot[\cdot]\colon T \times \mathbb{N} \to T \cup \{0\} \quad (\text{we assume } 0 \notin T)
\end{aligned}$$

such that

$$\forall a{\in}T \forall n{\in}\mathbb{N} (a[n] \in T \leftrightarrow n < \mathrm{arity}(\mathrm{rl}(a))).$$

A notation system $\mathcal{T} = (T, \mathrm{tp}, \mathrm{fv}, \mathrm{rl}, \cdot[\cdot])$ is *correct* if for every notation a and its predecessors the assigned types and sets of free variables satisfy some obvious conditions. An *α-norm* for a notation system $\mathcal{T}$ is a function o from T into the ordinals $<\alpha$ such that for all $a \in T$ and all $n < \mathrm{arity}(\mathrm{rl}(a))$ we have $o(a[n]) < o(a)$. A *rank* function for a notation system $\mathcal{T}$ is a function $\mathrm{rk}\colon T \to \mathbb{N}$ from T into the natural numbers such that (i) for all $a \in T$ and all $n < \mathrm{arity}(\mathrm{rl}(a))$ we have $\mathrm{rk}(a[n]) \leq \mathrm{rk}(a)$, and also (ii) $\mathrm{lev}(\mathrm{tp}(a[0])) \leq \mathrm{rk}(a)$ if $\mathrm{rl}(a)$ is a 'critical' application, i.e. not an application starting with a variable or a sequence.

7.7. *The extension $\mathcal{T}^*$ of a notation system $\mathcal{T}$.* Let $\mathcal{T} = (T, \mathrm{tp}, \mathrm{fv}, \mathrm{rl}, \cdot[\cdot])$ be a notation system. We extend $\mathcal{T}$ by adding for every variable y a constant a_y, for every $n \geq 1$ an $n+2$-ary function symbol sub for substitution (written $\mathrm{sub}_{\vec{x}}(a, \vec{b})$, where $\vec{x}$ is a (code for a) list of n distinct variables $x_1, \ldots, x_n$), and also a binary function symbol β for beta-conversion and a unary function symbol red for reduction. Let T^* be the closure of T generated by these function symbols. Using some obvious coding machinery we may clearly assume that T^* is a primitive recursive subset of

$\mathbb{N}$. The main point is that we can now extend the functions tp, fv, rl and $\cdot[\cdot]$ from T to T^*, in a primitive recursive way. It is here where the insertion of repetition rules in 7.5 is used. One can show

- If $\mathcal{T}$ is a correct notation system, then so is $\mathcal{T}^*$.
- Every norm on $\mathcal{T}$ can be extended to a norm on $\mathcal{T}^*$ by defining

$$\begin{aligned} o(a_y) &:= 0,\\ o(\mathrm{sub}_{\vec{x}}(a,\vec{b})) &:= (\max o(\vec{b})) + o(a),\\ o(\beta(a,b)) &:= o(b) + o(a) + 1,\\ o(\mathrm{red}(a)) &:= 4^{o(a)}. \end{aligned}$$

- Every rank function on $\mathcal{T}$ can be extended to a rank function on $\mathcal{T}^*$ by defining

$$\begin{aligned} \mathrm{rk}(a_y) &:= 0,\\ \mathrm{rk}(\mathrm{sub}_{\vec{x}}(a,\vec{b})) &:= \max(\mathrm{lev}(\vec{\rho}^{\circ}), \mathrm{rk}(\vec{b}), \mathrm{rk}(a)), \quad \text{where } \vec{\rho}^{\circ} \text{ consists of those}\\ &\qquad \rho_i \text{ from } \vec{\rho} := \mathrm{Tp}(\vec{x}) \text{ with } \mathrm{rl}(b_i) \text{ an abstraction or } \textsc{Rep},\\ \mathrm{rk}(\beta(a,b)) &:= \max(\mathrm{lev}(\mathrm{tp}(b)), \mathrm{rk}(a), \mathrm{rk}(b)),\\ \mathrm{rk}(\mathrm{red}(a)) &:= \mathrm{rk}(a) \dot{-} 1. \end{aligned}$$

7.8. *Embedding of* $\textsc{Pcf}_\alpha$. The next step is to translate the $\textsc{Pcf}_\alpha$-terms into infinite terms; this is done essentially by Tait's method, as in 7.3. We obtain a correct notation system $\mathcal{P}$ for $\textsc{Pcf}_\alpha$-terms with primitive recursive rank function and $\alpha\cdot\omega$-norm.

As decribed in 7.7, we can now extend $\mathcal{P}$ to a correct notation system $\mathcal{P}^* = (P^*, \dots)$ with primitive recursive rank function $\mathrm{rk}\colon P^* \to \mathbb{N}$ and norm $o\colon P^* \to \varepsilon(\alpha)$, where $\varepsilon(\alpha)$ is the least ε-number bigger than α.

7.9. *Valuation functionals.* Consider a notation system $\mathcal{P}$ with α-norm o. Let $a \in P$ be a notation of a closed term t with type ρ. Then we want to define the value $\textsc{Val}_\rho(a)$ of t. However, for the definition of the $\textsc{Val}_\rho$ it is necessary to allow free variables in t. Therefore, we introduce an additional argument w which serves to code an assignment $\vec{x} \mapsto \vec{u}$ for these variables.

So before we can give a definition of the $\textsc{Val}_\rho$, we have to introduce such codes for assignments. Since we are interested ultimately in closed terms and have to consider free variables only for their inductive construction, it is sufficient to restrict ourselves to terms whose free variables have types from an appropriate finite set $S = \{\tau_1, \dots, \tau_m\}$. So let an assignment of functionals $u_1, \dots, u_m$ to variables $x_1, \dots, x_m$ be given, both of types $\tau_1, \dots, \tau_m$. Let $a_1, \dots, a_m \in P$ be notations for $x_1, \dots, x_m$ (i.e. $\mathrm{rl}(a_i) = \textsc{Var}_{x_i}$). First, transform all the u_i to a common type $\tau = \tau_S$ by means of transformation functionals $\mathrm{Tr}^{\tau}_{\tau_i}$ with inverses $\mathrm{Tr}^{\tau_i}_{\tau}$ (i.e.

$\mathrm{Tr}_{\tau}^{\tau_i}(\mathrm{Tr}_{\tau_i}^{\tau}(u)) = u)$. With these $\mathrm{Tr}_{\tau_i}^{\tau}(u_i)$ build a function w of type $\iota \to \tau$ such that $w(a_i) = \mathrm{Tr}_{\tau_i}^{\tau}(u_i)$. From w one can then read off the functionals u_i assigned to x_i by $u_i = \mathrm{Tr}_{\tau}^{\tau_i}(w(a_i))$. With these tools one can define VAL_{ρ} ($\rho \in S$) with respect to a fixed finite set S of types and our given notation system $\mathcal{P}$ with α-norm o. Note that for $a \in P$ with norm $o(a) < \beta \leq \alpha$ we have a definition by simultaneous β-recursion of the finitely many functionals VAL_{ρ} for $\rho \in S$, where all auxiliary functionals (in particular $\cdot[\cdot]$) are primitive recursive.

7.10. *Second proof of the trade-off theorem.* We now can assemble the tools discussed and give a second proof of the trade-off theorem in 7.2. However, due to the slightly weaker estimates in 7.5 we obtain $4_m(\alpha \cdot \omega)$ as a bound for β (not $2_{m+1}(\alpha + \omega)$ as above).

The proof is as follows. By 6.7 we may assume that every $\mathrm{Y}_{\sigma,\alpha}$ in M appears in a context $\mathrm{Y}_{\sigma,\alpha}N$ with closed N, and that M is β-normal. Construct a correct notation system $\mathcal{P}$ for PCF_{α} as in 7.8, and also a primitive recursive rank function and an $\alpha{\cdot}\omega$-norm for $\mathcal{P}$. Now extend $\mathcal{P}$ to a correct notation system $\mathcal{P}^*$, as described above, again with a primitive recursive rank function and an $\varepsilon(\alpha)$-norm. In $\mathcal{P}^*$ we can form $\mathrm{red}^m(M)$. Since we have $\mathrm{rk}(M) = \mathrm{lev}(\rho) \leq n + m + 1$ in $\mathcal{P}$, we know that $\mathrm{rk}(\mathrm{red}^m(M)) \leq n + 1$ in $\mathcal{P}^*$, hence all bound variables in the term denoted by $\mathrm{red}^m(M)$ have levels $\leq n$. Concerning the norm in $\mathcal{P}$ we have $o(M) < \alpha \cdot k < \alpha \cdot \omega$ for some $k < \omega$, hence in $\mathcal{P}^*$

$$o(\mathrm{red}^m(M)) < 4_m(\alpha \cdot k) =: \beta < 4_m(\alpha \cdot \omega).$$

Let S be the finite set of all types used in the term M, and $S_k := \{\, \rho \in S \mid \mathrm{lev}(\rho) \leq k \,\}$. As described in 7.9 we can now define $\mathrm{VAL}_{\rho}(a, w)$ for $\rho \in S_{n+1}$, with $a \in P^*$ of norm $o(a) < \beta$ and w of type $\tau := \tau_{S_n}$, by simultaneous β-recursion. But

$$M' := \mathrm{VAL}_{\mathrm{Tp}(M)}(\mathrm{red}^m(M), w_0),$$

– where w_0 codes the empty assignment – denotes the same functional as M. Hence we have found the closed PCF_{β}-term of we have looked for. □

References

[1] W. Ackermann. Zur Widerspruchsfreiheit der Zahlentheorie. *Math. Annalen*, 117:162–194, 1940.

[2] Holger Benl, Ulrich Berger, Helmut Schwichtenberg, Monika Seisenberger, and Wolfgang Zuber. Proof theory at work: Program development in the Minlog system. Submitted to Automated Deduction – A Basis for Applications (eds W. Bibel and P. Schmitt), Volume I: Foundations, Kluwer Academic Publishers, 1998.

[3] Ulrich Berger. *Totale Objekte und Mengen in der Bereichstheorie*. PhD thesis, Mathematisches Institut der Universität München, 1990.

[4] Ulrich Berger. Total sets and objects in domain theory. *Annals of Pure and Applied Logic*, 60:91–117, 1993.

[5] Wilfried Buchholz. Notation systems for infinitary derivations. *Archive for Mathematical Logic*, 30:277–296, 1991.

[6] Wilfried Buchholz, Adam Cichon, and Andreas Weiermann. A uniform approach to fundamental sequences and hierarchies. *Mathematical Logic Quarterly*, 40:273–286, 1994.

[7] Wilfried Buchholz and Stan Wainer. Provably computable functions and the fast growing hierarchy. In *Logic and Combinatorics*, volume 65 of *AMS-series Contemporary Mathematics*, pages 179–198, 1987.

[8] Peter Clote. Computation models and function algebras. Technical Report BCCS–95–01, Boston College, Computer Science Department, Boston, January 1995. To be publ. in 'Handbook of Recursion Theory' ed. by Ed Griffor.

[9] A. Cobham. The intrinsic computational difficulty of functions. In Y. Bar-Hillel, editor, *Logic, Methodology and Philosophy of Science II*, pages 24–30. North–Holland, Amsterdam, 1965.

[10] Stephen A. Cook. Computability and complexity of higher type functions. In Y.N. Moschovakis, editor, *Logic from Computer Science, Proceedings of a Workshop held November 13–17, 1989*, number 21 in MSRI Publications, pages 51–72. Springer Verlag, Berlin, Heidelberg, New York, 1992.

[11] Walter Dosch. Reduction relations in strict applicative languages. In *ISTCS '92*, volume 601 of *Lecture Notes in Computer Science*, pages 55–66. Springer Verlag, Berlin, Heidelberg, New York, 1992.

[12] Yuri L. Ershov. Computable functionals of finite types. *Algebra i Logika*, 11(4):367–437, 1972.

[13] Yuri L. Ershov. Maximal and everywhere defined functionals. *Algebra i Logika*, 13(4):374–397, 1974.

[14] Yuri L. Ershov. Model C of partial continuous functionals. In R. Gandy and M. Hyland, editors, *Logic Colloquium 1976*, pages 455–467. North–Holland, Amsterdam, 1977.

[15] Matthew V.H. Fairtlough and Stanley S. Wainer. Hierarchies of provably recursive functions. to appear in: Handbook of Proof Theory (ed. Sam Buss), 1996.

[16] Steven Fortune, Daniel Leivant, and Michael O'Donnell. The expressiveness of simple and second–order type structures. *Journal of the Association for Computing Machinery*, 30(1):151–185, January 1983.

[17] Harvey Friedman and Michael Sheard. Elementary descent recursion and proof theory. *Annals of Pure and Applied Logic*, 71:1–45, 1995.

[18] Robin O. Gandy and J.M.E. Hyland. Computable and recursively countable functions of higher type. In *Logic Colloquium 76*, pages 407–438. North–Holland, Amsterdam, 1977.

[19] Jean-Yves Girard. Π^1_2–logic. Part I: Dilators. *Annals of Mathematical Logic*, 21:75–219, 1981.

[20] Kurt Gödel. Über eine bisher noch nicht benützte Erweiterung des finiten Standpunkts. *Dialectica*, 12:280–287, 1958.

[21] A. Grzegorczyk. *Some classes of recursive functions*, volume IV. Rozprawy Matematyczne, Warszawa, 1953.

[22] Hardy. A theorem concerning the infinite cardinal numbers. *Quaterly Journal of Mathematics*, 35:87–94, 1904.

[23] Peter G. Hinman. *Recursion–Theoretic Hierarchies*. Springer, Berlin, 1978.

[24] H. Rogers Jr. *Theory of recursive functions and effective computability*. Mc Graw Hill, 1967.

[25] Stephen C. Kleene. Extension of an effectively generated class of functions by enumeration. *Colloquium Mathematicum*, 7:67–78, 1958.

[26] Stephen C. Kleene. Countable functionals. In A. Heyting, editor, *Constructivity in Mathematics*, pages 81–100. North–Holland, Amsterdam, 1959.

[27] Stephen C. Kleene. Recursive functionals and quantifiers of finite types I. *Trans. Amer. Math. Soc*, 91:1–52, 1959.

[28] Stephen C. Kleene. Turing machine computable functionals of finite types I. In E. Nagel et al., editor, *Proc. of the Congress for Logic, Methodology and the Philosophy of Science*, pages 38–45, Stanford, 1962.

[29] Georg Kreisel. On the interpretation of non–finitist proofs II. *The Journal of Symbolic Logic*, 17:43–58, 1952.

[30] Georg Kreisel. Interpretation of analysis by means of constructive functionals of finite types. In A. Heyting, editor, *Constructivity in Mathematics*, pages 101–128. North–Holland, Amsterdam, 1959.

[31] K.G. Larsen and G. Winskel. Using information systems to solve recursive domain equations. *Information and Computation*, 91:232–258, 1991.

[32] M.H. Lob and Stanley S. Wainer. Hierarchies of number–theoretic functions I. *Archiv für Mathematische Logik und Grundlagenforschung*, 13:39–51, 1970.

[33] M.H. Lob and Stanley S. Wainer. Hierarchies of number–theoretic functions II. *Archiv für Mathematische Logik und Grundlagenforschung*, 13:97–113, 1970.

[34] M.L. Minsky. Recursive unsolvability of Post's problem of "Tag" and other topics in the theory of Turing machines. *Annals of Mathematics*, 74:437–455, 1961.

[35] Grigorii E. Mints. Quantifier–free and one–quantifier systems. *Journal of Soviet Mathematics*, 1:71–84, 1973.

[36] Grigorii E. Mints. Finite investigations of transfinite derivations. *Journal of Soviet Mathematics*, 10:548–596, 1978. Translated from: Zap. Nauchn. Semin. LOMI 49 (1975).

[37] Grigorii E. Mints. *Selected Papers in Proof Theory.* Studies in Proof Theory. Bibliopolis and North–Holland, Napoli and Amsterdam, 1992.

[38] J. Moldestad. *Computations in Higher Types*, volume 811 of *Lecture Notes in Mathematics*. Springer Verlag, Berlin, Heidelberg, New York, 1977.

[39] John Myhill. A stumblingblock in constructive mathematics (abstract). *The Journal of Symbolic Logic*, 18:190, 1953.

[40] Karl-Heinz Niggl. Subrecursive hierarchies on Scott domains. *Archive for Mathematical Logic*, 32:239–257, 1993.

[41] Karl-Heinz Niggl. *Subrecursive hierarchies on the partial continuous functionals.* PhD thesis, Universität München, Fakultät für Mathematik, 1994.

[42] Karl-Heinz Niggl. Towards the computational complexity of PR^ω–terms. *Annals of Pure and Applied Logic*, 75:153–178, 1995.

[43] Dag Normann. The continuous functionals. to appear: Handbook of Recursion Theory (ed. E. Griffor), 1996.

[44] Rósza Péter. *Rekursive Funktionen.* Akademie–Verlag, Berlin, 1957.

[45] Richard A. Platek. *Foundations of recursion theory.* PhD thesis, Department of Mathematics, Stanford University, 1966.

[46] Gordon D. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5:223–255, 1977.

[47] Joel W. Robbin. *Subrecursive Hierarchies.* PhD thesis, Princeton University, 1965.

[48] Harvey E. Rose. *Subrecursion: Functions and hierarchies*, volume 9 of *Oxford Logic Guides*. Clarendon Press, Oxford, 1984.

[49] N.A. Routledge. Ordinal recursion. *Proc. Cambridge Phil. Soc.*, 49:175–182, 1953.

[50] Gerald E. Sacks. *Higher Recursion Theory.* Perspectives in Mathematical Logic. Springer Verlag, Berlin, Heidelberg, New York, 1990.

[51] Kurt Schütte. Beweistheoretische Erfassung der unendlichen Induktion in der Zahlentheorie. *Mathematische Annalen*, 122:369–389, 1951.

[52] Helmut Schwichtenberg. Rekursionszahlen und die Grzegorczyk–Hierarchie. *Archiv für Mathematische Logik und Grundlagenforschung*, 12:85–97, 1969.

[53] Helmut Schwichtenberg. Eine Klassifikation der ϵ_0–rekursiven Funktionen. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 17:61–74, 1971.

[54] Helmut Schwichtenberg. Beweistheoretische Charakterisierung einer Erweiterung der Grzegorczyk–Hierarchie. *Archiv für Mathematische Logik und Grundlagenforschung*, 15:129–145, 1972.

[55] Helmut Schwichtenberg. Einige Anwendungen von unendlichen Termen und Wertfunktionalen. Habilitationsschrift, Mathematisches Institut der Universität Münster, 1973.

[56] Helmut Schwichtenberg. Elimination of higher type levels in definitions of primitive recursive functionals by means of transfinite recursion. In H.E. Rose and J.C. Shepherdson, editors, *Logic Colloquium '73*, pages 279–303. North–Holland, Amsterdam, 1975.

[57] Helmut Schwichtenberg. Proof theory: Some applications of cut–elimination. In J. Barwise, editor, *Handbook of Mathematical Logic*, volume 90 of *Studies in Logic and the Foundations of Mathematics*, chapter Proof Theory and Constructive Mathematics, pages 867–895. North–Holland, Amsterdam, 1977.

[58] Helmut Schwichtenberg and Stanley S. Wainer. Ordinal bounds for programs. In P. Clote and J. Remmel, editors, *Feasible Mathematics II*, pages 387–406. Birkhäuser, Boston, 1995.

[59] Dana Scott. Outline of a mathematical theory of computation. Technical Monograph PRG–2, Oxford University Computing Laboratory, 1970.

[60] Dana Scott. Domains for denotational semantics. In E. Nielsen and E.M. Schmidt, editors, *Automata, Languages and Programming*, volume 140 of *Lecture Notes in Computer Science*, pages 577–613. Springer Verlag, Berlin, Heidelberg, New York, 1982. A corrected and expanded version of a paper prepared for ICALP'82, Aarhus, Denmark.

[61] J.C. Shepherdson and H.E. Sturgis. Computability of recursive functions. *J. Ass. Computing Machinery*, 10:217–255, 1963.

[62] Joseph R. Shoenfield. *Mathematical Logic.* Addison–Wesley Publ. Comp., Reading, Massachusetts, 1967.

[63] Viggo Stoltenberg-Hansen, Edward Griffor, and Ingrid Lindström. *Mathematical Theory of Domains.* Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1994.

[64] William W. Tait. Nested recursion. *Math. Annalen*, 143:236–250, 1961.

[65] William W. Tait. Infinitely long terms of transfinite type. In J. Crossley and M. Dummett, editors, *Formal Systems and Recursive Functions*, pages 176–185. North–Holland, Amsterdam, 1965.

[66] William W. Tait. Normal derivability in classical logic. In J. Barwise, editor, *The Syntax and Semantics of Infinitary Languages*, volume 72 of *Lecture Notes in Mathematics*, pages 204–236. Springer Verlag, Berlin, Heidelberg, New York, 1968.

[67] Gaisi Takeuti. *Proof Theory.* North–Holland, Amsterdam, second edition, 1987.

[68] Gaisi Takeuti. Grzegorcyk's hierarchy and $iep\sigma_1$. *The Journal of Symbolic Logic*, 59(4):1274–1284, 1994.

[69] Stanley S. Wainer. A classification of the ordinal recursive functions. *Archiv für Mathematische Logik und Grundlagenforschung*, 13:136–153, 1970.

[70] Andreas Weiermann. Investigations on slow versus fast growing: how to majorize slow growing functions nontrivially by fast growing ones. *Archive for Mathematical Logic*, 34:313–330, 1995.

[71] Fred Zemke. P.R.–regulated systems of notation and the subrecursive hierarchy equivalence property. *Transactions of the American Math. Soc.*, 234:89–118, 1977.