Ordinal Bounds for Programs

Helmut Schwichtenberg   Stanley S. Wainer *

Introduction

In this paper we use methods of proof theory to assign ordinals to classes of (terminating) programs, the idea being that the ordinal assignment provides a uniform way of measuring computational complexity "in the large". We are not concerned with placing prior (e.g. polynomial) bounds on computation–length, but rather with general methods of assessing the complexity of natural classes of programs according to the ways in which they are constructed. We begin with an overview of the method in section 2, the crucial idea being supplied by Buchholz's ω^+–rule. Section 3 introduces a large class of higher–order programs based on Plotkin's PCF, but with "bounded" fixed point operators controlled by given well–orderings. A Tait–style computability proof then ensures termination. In section 4 the details of the ordinal assignment method are worked out for the case where the given well–ordering is just the ordering of the natural numbers. The complexity bounds thus obtained turn out to be the slow–growing functions G_α with α below the Bachmann–Howard ordinal. Thus the functions computed by PCF$_<$–programs are just the provably recursive functions of arithmetic.

1 Ordinal complexity of programs

Imagine a program P as a transition relation

$$\longrightarrow_P \subseteq \; (\text{States} \times \text{Data})^2$$

so that a terminating computation of P, starting in initial state s_0 with input a_0, is a sequence

$$(s_0, a_0) \longrightarrow_P (s_1, a_1) \longrightarrow_P \ldots \longrightarrow_P (s_k, a_k)$$

where s_k is a terminal state and a_k the output.

A complexity–bound for the program is then a function $g_P : N \to N$ such that on inputs a_0 of size $\leq n$ the length k of the ensuing computation is $\leq g_P(n)$.

Both authors partially supported by the Science Twinning Contract SC1–CT91–0724 of the European Community.

If one wishes to compare and classify the complexities of programs in a given language L, one would like (if possible) to arrange a suitable collection of bounding functions g_P in a linear, well-founded scale. That is, one would try to associate with L a "natural" and "minimal" well–ordering $||L||$, together with a hierarchy of functions $\{G_\alpha : \alpha \in ||L||\}$, and an ordinal assignment $P \in L \longrightarrow \alpha_P \in ||L||$ such that

(i) G_{α_P} is a complexity bound for P,

(ii) if $\alpha < \beta \in ||L||$ then G_β dominates G_α.

The aim here is to show how this can be achieved, at least for certain natural classes of terminating programs.

The methods are those of proof theory and the origins lie in [5, 3, 4, 11, 8]. First we need to develop a little "constructive–ordinal–theory".

1.1. Definition. The set Ω_1 of countable *tree–ordinals* is generated inductively by three rules:

$$0 \in \Omega_1,$$
$$\alpha \in \Omega_1 \implies \alpha + 1 \in \Omega_1,$$
$$\forall x \in N(\alpha_x \in \Omega_1) \implies \alpha = \langle \alpha_x \rangle \in \Omega_1.$$

Alternatively we can define Ω_1 in terms of Martin–Löf's well–founded type constructor W:

$$\Omega_1 \equiv Wx \in N_3.R_3(x, \perp, \top, N).$$

1.2. Definition. The sub–tree partial ordering $<$ on Ω_1 is just the transitive closure of

$$\alpha < \alpha + 1 \qquad \text{and} \qquad \forall n \in N(\alpha_n < \langle \alpha_x \rangle).$$

1.3. Definition. For each $\alpha \in \Omega_1$ and each $n \in N$ define the finite set $\alpha[n]$ of sub–trees of α by: $0[n] = \emptyset$, $(\alpha+1)[n] = \alpha[n] \cup \{\alpha\}$, $\langle \alpha_x \rangle[n] = \alpha_n[n]$.

Then $\alpha[n]$ is easily seen to be linearly ordered by $<$.

1.4. Definition. $\alpha \in \Omega_1$ is said to be *structured* if it is presented as the directed union of its finite subsets $\alpha[n]$ as follows:

(i) $\alpha[0] \subseteq \alpha[1] \subseteq \alpha[2] \subseteq \dots$

(ii) $\beta < \alpha \iff \exists n \in N(\beta \in \alpha[n])$.

Thus if α is structured, $\{\beta : \beta < \alpha\}$ is *well–ordered* with least element 0 and such that $\beta < \alpha \implies \beta + 1 \leq \alpha$.

1.5. Remark. Although we have not placed any effectivity conditions on the formation of sequences $\langle \alpha_x \rangle \in \Omega_1$, the particular examples of tree–ordinals that we construct here will all be recursive, and also structured. Thus when we construct and "name" a particular $\alpha \in \Omega_1$, it will actually code a recursive well–ordering.

For example any primitive recursive definition of a number–theoretic function f, say

$$f(a,0) = g_0(a),$$
$$f(a,b+1) = g_1(a,b,f(a,b))$$

can be lifted in a formally "continuous" way, to a primitive recursion over Ω_1 thus

$$\varphi(\alpha,0) = \psi_0(\alpha),$$
$$\varphi(\alpha,\beta+1) = \psi_1(\alpha,\beta,\varphi(\alpha,\beta)),$$
$$\varphi(\alpha,\langle\beta_x\rangle) = \langle\varphi(\alpha,\beta_x)\rangle.$$

This automatically gives rise to the obvious definitions of addition, multiplication and exponentiation over Ω_1. So starting with $x = 0+1+1\ldots+1$ (x times) we can define $\omega \in \Omega_1$ by the identity sequence $\omega = \langle x \rangle$ and then generate all Cantor forms in Ω_1 by $\alpha = \omega^{\alpha_1}\cdot n_1 + \omega^{\alpha_2}\cdot n_2 + \ldots + \omega^{\alpha_k}\cdot n_k$ etcetera.

1.6. Definition (Ordinal Assignment). Write $n{:}\, N \models^\alpha P\downarrow$ to mean that for every input a_0 of size $\leq n$ there is a terminating computation sequence

$$(s_0,a_0) \longrightarrow_P (s_1,a_1) \longrightarrow_P \ldots \longrightarrow_P (s_k,a_k)$$

and an assignment of a tree–ordinal $\beta_i \in \alpha[n]$ to each step (s_i,a_i), $i<k$, such that $\beta_0 < \beta_1 < \ldots < \alpha$.

Finally write $\models^\alpha P\downarrow$ to mean $\forall n(n{:}\, N \models^\alpha P\downarrow)$.

Note. In order to have $n{:}\, N \models^\alpha P\downarrow$ it is only necessary that the size of $\alpha[n]$ be at least as big as the maximum "computation length" of P, on inputs of size $\leq n$. However in order to have $\models^\alpha P\downarrow$ we need to construct α such that $n{:}\, N \models^\alpha P\downarrow$ *uniformly* over all n. The following is immediate.

1.7. First Collapsing Lemma. *Let $G_\alpha(n)$ = size of $\alpha[n]$. Then for all programs P,*

$$\models^\alpha P\downarrow \iff G_\alpha \textit{ is a complexity bound for } P.$$

Note. It is obvious from the definition of $\alpha[n]$ that

$$G_0(n) = 0, \quad G_{\alpha+1}(n) = G_\alpha(n)+1, \quad G_{\langle\alpha_x\rangle}(n) = G_{\alpha_n}(n).$$

G_α is the so–called Slow Growing Hierarchy. Note that everything is measured here "pointwise" with respect to the size n of inputs. Note also that the function G_α depends crucially on the way in which a structured tree-ordinal α is presented as the directed union of its finite subsets $\alpha[n]$, and not just on its set-theoretic ordinal rank $|\alpha|$. Simply knowing that $|\alpha| \leq |\beta|$ gives no information whatever about the relative sizes of G_α and G_β, since for example, if f is an arbitrary increasing number-theoretic function we can define the structured tree-ordinal $\alpha_f = \langle f(x) \rangle$ with ordinal rank ω such that $G_{\alpha_f} = f$. However, if instead we require the existence of a "natural transformation" from α to β, that is an order-preserving map π from $\{\gamma : \gamma < \alpha\}$ into $\{\gamma : \gamma < \beta\}$ such that for every n, π takes $\alpha[n]$ into $\beta[n]$, then we do obtain $\forall n\,(G_\alpha(n) \leq G_\beta(n))$. Thus proof-theoretic ordinal analysis is about the ways in which ordinals are presented (ordinal notations), and not about set-theoretic ordinals. What we are interested in here, is the development of a fairly general framework for assigning to a given class L of terminating programs, a single structured tree- ordinal τ such that the hierarchy of functions $\{G_\alpha : \alpha < \tau\}$ is a well-ordered scale of complexity bounds for the programs in L.

Suppose now that we have some language L within which we can generate terminating programs, by starting with some given "basic" programs B (for example zero, successor) and then building new programs by composition (sequencing) and applications of certain given "program–constructors" C (for example loops or recursors).

Our standing assumption on basic programs will be that they operate in one step and increase the size of the input by at most one.

Each constructor C is a uniform method for constructing new programs $C(Q)$ from previously generated ones Q. In terms of the formalism already developed, its uniformity can be measured by a sequence $\langle \alpha_\xi \rangle$ such that

$$\forall \xi \forall Q (\models^{\xi} Q \downarrow \Longrightarrow \models^{\alpha_\xi} C(Q) \downarrow).$$

Thus $\alpha = \langle \alpha_\xi \rangle$ measures the complexity of C itself. But unfortunately α is uncountable and therefore no longer a member of Ω_1. Therefore if we are to make ordinal assignments to programs in L, it is quite natural to extend Ω_1 to the next higher number class $\Omega_2 \equiv Wx \in N_4.R_4(x, \perp, \top, N, \Omega_1)$ generated inductively by the four rules:

$$\begin{aligned}
&0 \in \Omega_2,\\
&\alpha \in \Omega_2 \implies \alpha + 1 \in \Omega_2,\\
&\forall x \in N(\alpha_x \in \Omega_2) \implies \alpha = \langle \alpha_x \rangle \in \Omega_2,\\
&\forall \xi \in \Omega_1(\alpha_\xi \in \Omega_2) \implies \alpha = \langle \alpha_\xi \rangle \in \Omega_2.
\end{aligned}$$

Then we can derive each program in L and simultaneously assign to it an ordinal bound in Ω_2, according to the following rules:

- *Basic.* $\vdash^1 B$

- *Composition.* If $\vdash^{\alpha_0} Q$ and $\vdash^{\alpha_1} P$ then $\vdash^{\alpha_0 + \alpha_1^*} P \circ Q$.

- *Constructors.* If $\forall\xi\forall Q(\models^{\xi} Q\downarrow \implies \vdash^{\alpha_\xi} C(Q))$ and $\vdash^{\alpha_0} P$ then $\vdash^{\alpha} C(P)$.

Note. In the composition rule the ordinal bound $\alpha_0+\alpha_1$ would obviously be the most natural and desirable. However, technical reasons concerning the proof of the second collapsing lemma below necessitate a modification to $\alpha_0+\alpha_1^*$ where $\alpha_1^*=\langle\gamma_\xi\rangle$ with $\gamma_0=\alpha_1$ and $\gamma_\xi=\xi$ on all $\xi\neq 0$. Clearly this is a rather strange, "non–monotone" sequence, but it satisfies exactly the right identity with respect to the collapsing function D defined below. Other more natural choices are possible, for example we could instead take $\alpha_1^*=\alpha_1+\langle\xi\rangle$.

Note. The uniformity clause on α in the constructor rule is predicated upon the assumption $\models^{\xi} Q\downarrow$ and not $\vdash^{\xi} Q$. The reason is that if $\models^{\xi} Q\downarrow$ were replaced by $\vdash^{\xi} Q$ the inductive definition would not be "positive" and would not make good sense. So we are forced to adopt two stages: first define $\models^{\xi}$ and then define $\vdash^{\alpha}$.

Note. The constructor–rule is nothing other than Buchholz's ω^+–Rule (cf. [5, p. 195], where it is called Ω–Rule), developed originally in his proof–theoretic analysis of inductive definitions, but it appears here in a somewhat simpler context. What follows is a fundamental idea in his analysis.

The point is that once we have derived a program P with uncountable ordinal bound $\alpha\in\Omega_2$

$$\vdash^{\alpha} P$$

we know that it ought to be possible to collapse α down to a countable ordinal bound $D\alpha\in\Omega_1$ with

$$\models^{D\alpha} P\downarrow .$$

The reason is simple — there can only be countably–many different computations of P, so there has to be a countable upper bound!

Knowing only that such a collapsing map $D:\Omega_2\to\Omega_1$ exists, we can more or less read off its definition from the above rules for $\vdash^{\alpha} P$. In fact we define first a binary version of $D:\Omega_2\times\Omega_1\to\Omega_1$. The second parameter β in $D(\alpha,\beta)$ is to be thought of as a bound on any potential input. The conditions on D are as follows:

- *Basic.* $D(1,\beta)=\beta+1$
- *Composition-0.* $D(\alpha_0+\alpha_1,\beta)=D(\alpha_1,D(\alpha_0,\beta))$
- *Composition-1.* $D(\alpha_0+\alpha_1^*,\beta)=D(\alpha_0,\beta)+D(\alpha_1,D(\alpha_0,\beta))$
- *Constructors.* $D(\alpha,\beta)=D(\alpha_{D(\alpha_0,\beta)},\beta)$

It is easy to check that the following recursive definition of D does actually satisfy these conditions:

$$\begin{aligned} D(0,\beta) &= \beta, \\ D(\alpha+1,\beta) &= D(\alpha,\beta)+1, \\ D(\alpha,\beta) &= \langle D(\alpha_x,\beta)\rangle \qquad \text{if } \alpha = \langle \alpha_x \rangle, \\ D(\alpha,\beta) &= D(\alpha_{D(\alpha_0,\beta)},\beta) \qquad \text{if } \alpha = \langle \alpha_\xi \rangle. \end{aligned}$$

If we now put $D\alpha = D(\alpha,\omega)$ then $D\alpha$ is exactly the collapsing function introduced by Buchholz in chapter IV of [5], page 195, with the slight difference that he starts with $D0 = 0$ rather than our $D0 = \omega$, and we obtain

1.8. Second Collapsing Lemma.

$$\vdash^{\alpha} P \implies \models^{D\alpha} P\downarrow.$$

Proof. We in fact need to prove something rather more general. What we shall show is the following. If $\vdash^\alpha P$ then for all n and for all $\beta \geq \omega$ we have

(1) $G_\beta(n)\colon N \models^{D(\alpha,\beta)} P\downarrow$ and

(2) for each input a of size $\leq G_\beta(n)$ the output Pa is of size $\leq G_{D(\alpha,\beta)}(n)$.

The result then follows immediately from (1) by putting $\beta = \omega$, since we take $\omega := \langle x \rangle$ and hence $G_\omega(n) = n$. We proceed by induction on the derivation $\vdash^\alpha P$. There are three cases.

Basic. Suppose $\vdash^1 B$ and let n and $\beta \geq \omega$ be given. The assumption on basic programs is that given an input of size $\leq G_\beta(n)$, the computation terminates in one step with output of size $\leq G_\beta(n)+1$. Therefore since $G_\beta(n)+1 = G_{\beta+1}(n)$ we have $G_\beta(n)\colon N \models^{\beta+1} B\downarrow$. But $D(1,\beta) = \beta+1$. Hence the result.

Composition. Suppose $\vdash^{\alpha_0+\alpha_1^*} P\circ Q$ arises from $\vdash^{\alpha_0} Q$ and $\vdash^{\alpha_1} P$ and let n and $\beta \geq \omega$ be given. By one application of the induction hypothesis

$$G_\beta(n)\colon N \models^{D(\alpha_0,\beta)} Q\downarrow$$

and on input a of size $\leq G_\beta(n)$ the output $b = Qa$ is of size $\leq G_{D(\alpha_0,\beta)}(n)$. Now we can use the induction hypothesis on $\vdash^{\alpha_1} P$ with β replaced by $D(\alpha_0,\beta)$ to obtain

$$G_{D(\alpha_0,\beta)}(n)\colon N \models^{D(\alpha_1,D(\alpha_0,\beta))} P\downarrow$$

with output Pb of size $\leq G_{D(\alpha_1,D(\alpha_0,\beta))}(n)$. Composing these we therefore have

$$G_\beta(n)\colon N \models^{D(\alpha_0,\beta)+D(\alpha_1,D(\alpha_0,\beta))} (P\circ Q)\downarrow.$$

The result then follows by property Composition–1 of the D–function.

Constructors. Suppose $\vdash^{\alpha} C(P)$ arises from $\vdash^{\alpha_0} P$ and $\forall\xi\forall Q(\models^{\xi} Q\downarrow \implies \vdash^{\alpha_\xi} C(Q))$, and let n and $\beta \geq \omega$ be given. Applying the induction hypothesis to $\vdash^{\alpha_0} P$ we obtain

$$G_\beta(n)\colon N \models^{D(\alpha_0,\beta)} P\downarrow$$

where, on inputs of size $\leq G_\beta(n)$, the output of P is of size $\leq G_{D(\alpha_0,\beta)}(n)$. In particular then $\models^{D(\alpha_0,\beta)} P\downarrow$ by definition. Now we can apply the other assumption with $\xi = D(\alpha_0,\beta)$ and $Q = P$ to obtain $\vdash^{\alpha_\xi} C(P)$. Therefore by another application of the induction hypothesis we have firstly $G_\beta(n)\colon N \models^{D(\alpha_\xi,\beta)} C(P)\downarrow$ and secondly, on inputs of size $\leq G_\beta(n)$, the output given by $C(P)$ is of size $\leq G_{D(\alpha_\xi,\beta)}(n)$. Since with $\xi = D(\alpha_0,\beta)$ we have $D(\alpha_\xi,\beta) = D(\alpha,\beta)$, this completes the proof. ∎

By combining the First and Second Collapsing Lemmas, we therefore have a way of reading off complexity bounds for programs in L:

$$\vdash^{\alpha} P \implies P \text{ has a complexity bound } G_{D\alpha}.$$

Of course the trick lies in finding appropriate uniform measures $\alpha = \langle\alpha_\xi\rangle$ for L–constructors. But once this is done, we have a method for classifying the computational complexity of L–programs in terms of the $G_{D\alpha}$–functions. These are well–known functions in subrecursive hierarchy theory — the so–called Fast Growing Functions — and they provide immediate access to proof–theoretic characterizations of the functions definable by L–programs.

What we have described here are only the conceptual ideas. Precise results require more detail concerning the structure of the language L. In the rest of the paper we work out these details for a natural class of higher–order programs based on Gödel's Dialectica system T, following [11]. We can however already see how this analysis should proceed. Gödel's T is based on the primitive recursion constructor C which, provided with a numerical input m, makes exactly m recursive calls on any given function parameter. Thus if Q computes the numerical input with ordinal-bound ξ then ξ should also be the bound on the derivation of C(Q). In other words the ordinal-sequence measuring this C should be the identity sequence $\Omega = \langle\xi\rangle$ in Ω_2. If k nested applications of C are used then the ordinal-bound will be $\Omega \cdot k$. But we are here working in a higher-order setting, so a Normalization Theorem will be needed to reduce a higher-order program with "cut- rank" n to a first-order one. This will increase the initial ordinal bound $\Omega \cdot k$ to $E^n(\Omega \cdot k)$ where E is the exponential function on Ω_2. Collapsing this to the countable tree-ordinal $DE^n(\Omega \cdot k)$ then yields the desired ordinal estimate for the given program. This is precisely the procedure taken by Buchholz in his brief analysis of ID_1 on pages 195-196 of [5], but of course the context and the intricate details are different here. Nevertheless we can begin to see that the upper bound on the ordinal measures for Gödel's T is going to be the same as the ordinal bound for ID_1, namely the Bachmann–Howard ordinal. However, much work is required in what follows, in order to set this up precisely.

2 PCF with bounded fixed points

We study restricted notions of computability in the context of the partial continuous functionals. In particular we introduce bounded versions of the fixed point operators Y_ρ, depending on a given well–ordering. The fixed point operator Y_ρ of type $(\rho \to \rho) \to \rho$ satisfies $\mathrm{Y}_\rho G = G(\mathrm{Y}_\rho G)$. For example, the factorial may be defined by

$$\mathrm{Y}_{\mathbf{nat}\to\mathbf{nat}}(\lambda f \lambda x.\mathrm{cond}(\mathrm{zerop}\ x)1(\mathrm{mult}\ x(f(\mathrm{pred}\ x)))).$$

The intention here is that $\mathrm{Y}Gx$ will only use its functional argument G in such a way that G itself calls its function argument at numbers "less" than x. We propose to make this intention explicit and restrict Y accordingly. So let a well–ordering $\prec$ be given. We want $\mathrm{Y}_{\rho,\prec}$ to satisfy

$$\mathrm{Y}_{\rho,\prec}Gx = G(\lambda y.\mathrm{cond}(y \prec x)(\mathrm{Y}_{\rho,\prec}Gy)\bot)x.$$

This equality, when viewed as a conversion rule from left to right, clearly gives rise to an infinite reduction sequence: just repeatedly expand the subterm $\mathrm{Y}_{\rho,\prec}Gy$ by the same rule again. Hence in order to achieve strong normalization we have to make use of the fact that the subterm $\mathrm{Y}_{\rho,\prec}Gy$ occurs in a "guarded" context $\mathrm{cond}(y \prec x)(\mathrm{Y}_{\rho,\prec}Gy)\bot$ only. Therefore we restrict our conversion rules and allow conversion of a term $\mathrm{cond}MNL$ only if M is a boolean constant (i.e. tt, ff or $\bot^{\mathbf{boole}}$). 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 any term 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 [9], 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 in [6]. However, Dosch only deals with strict functionals.

For the particular case $<$ (the standard ordering of the natural numbers) $\mathrm{Y}_{\rho,<}$ is definable from the standard higher order recursion operators R_ρ, adapted to the partial continuous functionals. We then show that any closed term M of type $\mathbf{nat} \to \mathbf{nat}$ involving only bounded fixed point operators $\mathrm{Y}_{\rho,<}$ has the property that values of the $M\mathbf{n}$ for $n = 0, 1, 2, \ldots$ (if defined) are eventually bounded by a function G_α of the slow–growing subrecursive hierarchy with α denoting an ordinal below the Bachmann–Howard ordinal. From this we can immediately bound the complexity of the program M as in the First Collapsing Lemma.

2.1 Syntax of PCF$_\prec$

Let us first review some syntactical issues. Simple *types* $\rho, \sigma, \tau \ldots$ are built from ground types **nat** and **boole** (denoted by ι) by *function type* formation $\rho \to \sigma$. The *level* $\mathrm{lev}(\rho)$ of a type ρ is defined inductively by

$$\begin{aligned} \mathrm{lev}(\iota) &:= 0 \qquad \text{for any ground type } \iota; \\ \mathrm{lev}(\rho \to \sigma) &:= \max(\mathrm{lev}(\rho)+1, \mathrm{lev}(\sigma)). \end{aligned}$$

The type constructor $\to$ is understood to be associative to the right, so $\rho \to \sigma \to \tau$ means $\rho \to (\sigma \to \tau)$. Note that any type ρ can be written uniquely in the form $\vec{\rho} \to \iota := \rho_1 \to \rho_2 \to \cdots \to \rho_n \to \iota$ with a ground type ι, and its level is $\max(\mathrm{lev}(\rho_1), \mathrm{lev}(\rho_2), \ldots, \mathrm{lev}(\rho_n)) + 1$.

Let $\prec$ be a given well–ordering of the natural numbers. A *term* of PCF$_\prec$ is built from typed *variables* $x^\rho, y^\rho, z^\rho, \ldots$ and the typed *constants* listed below by *application* $M^{\rho\to\sigma}N^\rho$ and λ*–abstraction* $\lambda x^\rho M^\sigma$. The constants are

(1) $\perp^{\mathbf{nat}}$ and 0 of type **nat**,

(2) $\perp^{\mathbf{boole}}$, tt and ff of type **boole**,

(3) the sequential conditional cond_ι of type $\mathbf{boole} \to \iota \to \iota \to \iota$, for each ground type ι,

(4) the parallel conditional pcond_ι of type $\mathbf{boole} \to \iota \to \iota \to \iota$, for each ground type ι,

(5) $c_\prec$ and $c_=$ of type $\mathbf{nat} \to \mathbf{nat} \to \mathbf{boole}$ and succ of type $\mathbf{nat} \to \mathbf{nat}$,

(6) for each type ρ, the constant $\mathrm{Y}_{\rho,\prec}$ of type $(\vec{\rho} \to \mathbf{nat}) \to (\rho \to \rho) \to \rho$.

Application is understood to be associative to the left, so MNL means $(MN)L$.

Note that in contrast to [9] we do need a constant for the undefined object $\emptyset$ in our ground domains. This is necessary, since we want any reduction sequence to terminate, possibly with a symbol indicating an error. The constants $\perp^{\mathbf{nat}}$ and $\perp^{\mathbf{boole}}$ serve as such error symbols.

The term $\mathrm{succ}(\mathrm{succ} \ldots (\mathrm{succ}\ 0) \ldots)$ with n occurrences of succ is called a numeral and is denoted by $\mathbf{n}$.

2.2 Denotational semantics

Let Cont_ρ be the domain of the partial continuous functionals of type ρ, in the sense of Scott and Ersov (cf. [13], [9] or Schwichtenberg's lecture notes [12]). For any constant c^ρ of the language PCF$_\prec$ we define its value $[\![c^\rho]\!] \in \mathrm{Cont}_\rho$, as follows.

$$[\![\perp^{\mathbf{nat}}]\!] := [\![\perp^{\mathbf{boole}}]\!] := \emptyset; \quad [\![\mathrm{tt}]\!] := \{\mathrm{true}\}; \quad [\![\mathrm{ff}]\!] := \{\mathrm{false}\}; \quad [\![0]\!] := \{0\}.$$

$$[\![\mathrm{cond}_\iota]\!]pxy := \begin{cases} x, & \text{if } p = \{\text{true}\}; \\ y, & \text{if } p = \{\text{false}\}; \\ \emptyset, & \text{if } p = \emptyset. \end{cases}$$

$$[\![\mathrm{pcond}_\iota]\!]pxy := \begin{cases} x, & \text{if } p = \{\text{true}\}; \\ y, & \text{if } p = \{\text{false}\}; \\ x, & \text{if } p = \emptyset \text{ and } x = y; \\ \emptyset, & \text{if } p = \emptyset \text{ and } x \neq y. \end{cases}$$

It is well–known (cf. [9], p.237) that $[\![\mathrm{cond}_\iota]\!]$ can be defined from $[\![\mathrm{pcond}_\iota]\!]$ by

$$[\![\mathrm{cond}_\iota]\!]pxy = [\![\mathrm{pcond}_\iota]\!]p([\![\mathrm{pcond}_\iota]\!]px\emptyset)([\![\mathrm{pcond}_\iota]\!]p\emptyset y).$$

However, we must keep the cond_ι as separate constants in our language, since they will be given a special status in our operational semantics, which makes it possible to use them as "guards" blocking infinite reduction sequences.

$$[\![c_\prec]\!]xy := \begin{cases} \{\text{true}\}, & \text{if } x = \{n\},\ y = \{m\} \text{ and } n \prec m; \\ \{\text{false}\}, & \text{if } x = \{n\},\ y = \{m\} \text{ and } n \not\prec m; \\ \emptyset, & \text{if } x = \emptyset \text{ or } y = \emptyset, \end{cases}$$

and similarly for $c_=$ and succ.

The value $[\![\mathrm{Y}_{\rho,\prec}]\!]$ is to be an object of type $(\vec{\rho} \to \mathbf{nat}) \to (\rho \to \rho) \to \rho$. So let objects μ of type $\vec{\rho} \to \mathbf{nat}$, G of type $\rho \to \rho$ and $\vec{x}$ of types $\vec{\rho}$ be given. We view μ as a measure function mapping any sequence $\vec{y}$ of objects into the domain of the well–ordering $\prec$, which is the set of natural numbers. We now define $[\![\mathrm{Y}_{\rho,\prec}]\!]\mu G\vec{x}$ by transfinite recursion on $\mu\vec{x}$, as follows. If $\mu\vec{x}$ is undefined, then $[\![\mathrm{Y}_{\rho,\prec}]\!]\mu G\vec{x}$ is undefined. Otherwise, assume that for all $\vec{y}$ with $\mu\vec{y} \prec \mu\vec{x}$ the value $[\![\mathrm{Y}_{\rho,\prec}]\!]\mu G\vec{y}$ is already defined. Then let

$$[\![\mathrm{Y}_{\rho,\prec}]\!]\mu G\vec{x} := Gf\vec{x}$$

where f is defined by

$$f\vec{y} := \begin{cases} [\![\mathrm{Y}_{\rho,\prec}]\!]\mu G\vec{y}, & \text{if } \mu\vec{y} \prec \mu\vec{x}; \\ \emptyset, & \text{otherwise.} \end{cases}$$

Note that all finite functionals from Cont_ρ are definable by terms of $\mathrm{PCF}_\prec$ not involving fixed point operators; this has been observed in [9], Lemma 4.5.

2.3 Operational semantics

The operational semantics is given by the following conversion rules.

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

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

(3) $\mathrm{Y}_{\rho,\prec} MN\vec{N} \to \mathrm{cond}_\iota(c_=(M\vec{N})(M\vec{N}))$
$(N(\lambda\vec{y}.\mathrm{cond}_\iota(c_\prec(M\vec{y})(M\vec{N}))(\mathrm{Y}_{\rho,\prec}MN\vec{y})\bot)\vec{N})\bot$.

(4) $\mathrm{cond}_\iota \mathtt{tt} MN \to M$; $\mathrm{cond}_\iota \mathtt{ff} MN \to N$; $\mathrm{cond}_\iota \bot MN \to \bot$.

(5) $\dfrac{M \to M'}{\mathrm{cond}_\iota M \to \mathrm{cond}_\iota M'}$.

(6) $\mathrm{pcond}_\iota \mathtt{tt} MN \to M$; $\mathrm{pcond}_\iota \mathtt{ff} MN \to N$; $\mathrm{pcond}_\iota MNN \to N$;
$\mathrm{pcond}_\iota \bot M \bot \to \bot$; $\mathrm{pcond}_\iota \bot\bot N \to \bot$; $\mathrm{pcond}_\iota M \mathtt{tt}\mathtt{ff} \to M$;
$\mathrm{pcond}_\iota \bot cd \to \bot$ if c and d are different constants or numerals.

(7) $\dfrac{M \to M'}{\mathrm{pcond}_\iota M \to \mathrm{pcond}_\iota M'}$ $\qquad$ $\dfrac{N \to N'}{\mathrm{pcond}_\iota MN \to \mathrm{pcond}_\iota MN'}$
$\dfrac{L \to L'}{\mathrm{pcond}_\iota MNL \to \mathrm{pcond}_\iota MNL'}$

(8) $c_\prec \mathbf{mn} \to \begin{cases} \mathtt{tt}, & \text{if } m \prec n; \\ \mathtt{ff}, & \text{otherwise} \end{cases}$; $c_\prec \bot N \to \bot$; $c_\prec M \bot \to \bot$; similarly for $c_=$.
$\mathrm{succ}\bot \to \bot$.

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

It might seem that the simpler rule

(3') $\mathrm{Y}_{\rho,\prec} MN\vec{N} \to N(\lambda\vec{y}.\mathrm{cond}_\iota(c_\prec(M\vec{y})(M\vec{N}))(\mathrm{Y}_{\rho,\prec}MN\vec{y})\bot)\vec{N}$

for the Y–operator would suffice. However, it does not in case $M\vec{N}$ evaluates to $\bot^{\mathbf{nat}}$. For then the outcome of the simplified rule (3') depends on the form of N and might be defined, which would be inconsistent with our denotational semantics.

Note that we do *not* have

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

for then following an application of rule (3) for the bounded fixed point operator we could easily construct an infinite reduction sequence.

2.4 Normalization via computability predicates

We now prove that any term M is strongly normalizable, i.e. that any reduction sequence starting with M terminates. Note that it is only in the range of the constants other than the sequential conditional cond_ι that we have a choice of where to reduce.

If M is a closed term of ground type, then it clearly follows from the form of our conversion rules the final term in any reduction sequence must be a constant or a numeral.

For the proof we use W.W. Tait's method of computability predicates. So for any 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 any 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 any type respecting substitution of closed computable terms for the free variables in M yields a computable term.

2.1. Theorem. *Any term M is computable.*

Proof. By induction on M.

x: Trivial.

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.

$\lambda x M$: 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 any 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.

$\mathrm{Y}_{\rho,\prec}$: Let $M, N, \vec{N}$ be closed computable terms. We have to show that $\mathrm{Y}_{\rho,\prec}MN\vec{N}$ is computable, i.e. that any reduction sequence starting with that term terminates. Since M is computable by assumption, we know that $M\vec{N}$ reduces to a constant or a numeral c; we use an auxiliary transfinite $\prec$–induction on (the value of) c. If $c = \bot$, then any reduction sequence starting with $\mathrm{Y}_{\rho,\prec}MN\vec{N}$ must after its first two steps work on reducing $c_=(M\vec{N})(M\vec{N})$ 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 $M\vec{N}$ reduces to $\bot$ we must get $\bot$, and hence $\mathrm{Y}_{\rho,\prec}MN\vec{N}$ reduces to $\bot$. So assume $c = \mathbf{n}$. Then as before we can see that any reduction sequence starting with $\mathrm{Y}_{\rho,\prec} MN\vec{N}$ must come across

$$N(\lambda\vec{y}.\mathrm{cond}_\iota(c_\prec(M\vec{y})(M\vec{N}))(\mathrm{Y}_{\rho,\prec} MN\vec{y})\bot)\vec{N}.$$

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

$$\lambda\vec{y}.\mathrm{cond}_\iota(c_\prec(M\vec{y})(M\vec{N}))(\mathrm{Y}_{\rho,\prec} MN\vec{y})\bot$$

is computable. So let $\vec{L}$ be a sequence of closed computable terms. We must show that

$$\mathrm{cond}_\iota(c_\prec(M\vec{L})(M\vec{N}))(\mathrm{Y}_{\rho,\prec} MN\vec{L})\bot$$

is computable. Since M is computable by assumption, we know that $M\vec{L}$ reduces to a constant or a numeral d. If $d = \bot$, then $c_\prec(M\vec{L})(M\vec{N})$ 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

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

If $m \prec n$ holds, then $c_\prec\mathbf{mn} \to \mathtt{tt}$ and hence the whole term must reduce to $\mathrm{Y}_{\rho,\prec} MN\vec{L}$, which by hypothesis of our auxiliary $\prec$–induction is computable. If $m \prec n$ does not hold, then $c_\prec\mathbf{mn} \to \mathtt{ff}$ and hence the whole term reduces to $\bot$.

cond_ι, pcond_ι: Let M, N, L be closed computable terms. We have to show that $\mathrm{cond}_\iota MNL$ and $\mathrm{pcond}_\iota MNL$ are computable, i.e. that any 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 $\mathrm{PCF}_\prec$ the argument is similar. ∎

3 Primitive recursion in higher types, $\mathrm{PCF}_<$

Here we restrict ourselves to the case where the well–ordering $\prec$ considered is the standard ordering $<$ of the natural numbers. In this case the bounded fixed point operator is definable by means of primitive recursion, as follows.

Define a functional H_ρ of type $(\vec{\rho} \to \mathbf{nat}) \to (\rho \to \rho) \to \mathbf{nat} \to \rho$ by primitive recursion, as in Gödel's Dialectica system T:

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

where K_ρ of type $(\vec{\rho} \to \mathbf{nat}) \to (\rho \to \rho) \to \mathbf{nat} \to \rho \to \rho$ is defined by

$$K_\rho\mu Gxz\vec{y} := [\![\mathrm{cond}_\iota]\!]([\![c_=]\!](\mu\vec{y})x)(Gz\vec{y})([\![\mathrm{cond}_\iota]\!]([\![c_<]\!](\mu\vec{y})x)(z\vec{y})\emptyset).$$

Then we have, by induction on n

$$H_\rho \mu G\{n\}\vec{y} = \begin{cases} [\![\mathrm{Y}_{\rho,<}]\!]\mu G\vec{y}, & \text{if } \mu\vec{y} < \{n\}; \\ \emptyset, & \text{otherwise.} \end{cases}$$

Hence $[\![\mathrm{Y}_{\rho,<}]\!]$ is explicitely definable from H_ρ, by

$$[\![\mathrm{Y}_{\rho,<}]\!]\mu G\vec{y} = H_\rho \mu G([\![\mathrm{succ}]\!](\mu\vec{y}))\vec{y}.$$

Thus $\mathrm{PCF}_<$ is nothing other than a partial version of Gödel's T. We now give details of an ordinal analysis of this system, following the ideas outlined in section 2.

We show that any closed term M of type $\mathbf{nat} \to \mathbf{nat}$ involving only bounded fixed point operators $\mathrm{Y}_{\rho,<}$ has the property that values of the $M\mathbf{n}$ for $n = 0, 1, 2, \dots$ (if defined) are eventually bounded by a function G_α of the slow–growing subrecursive hierarchy with α denoting an ordinal below the Bachmann–Howard ordinal. The argument is similar to the one in [11], and we rely heavily on that paper.

By the remark above it suffices to deal with terms involving the recursion operators R_ρ instead of the $Y_{\rho,<}$'s. R_ρ has type $\rho \to (\mathbf{nat} \to \rho \to \rho) \to \mathbf{nat} \to \rho$, and the conversion rules associated with it are

$$\begin{aligned} R_\rho PQ0\vec{L} &\to P\vec{L}, \\ R_\rho PQ(\mathrm{succ}M)\vec{L} &\to QM(R_\rho PQM)\vec{L}. \end{aligned}$$

Since any such term has a unique $\beta\eta R$–normal form, we can identify $\beta\eta R$–equivalent terms.

For any term M let $\mathrm{lev}(M)$ denote the level of the type of M. For terms M with $\mathrm{lev}(M) = 0$, tree ordinals $\alpha \in \Omega_2$ and $n \in \mathbf{N}$ we now follow the ideas of section 2 in defining inductively the relation $\vdash_n^\alpha M$, to be read M is *generated* with size α and rank n, by the rules:

- *Variable Rule.* If $\vdash_n^\alpha L_i\vec{y_i}$ for $i = 1, \dots, m$ with $m \geq 0$, then $\vdash_n^{\alpha+1} xL_1 \dots L_m$.
- *Constant Rules.* $\vdash_n^{\alpha+1} \bot^{\mathbf{nat}}$, $\vdash_n^{\alpha+1} 0$, $\vdash_n^{\alpha+1} \bot^{\mathbf{boole}}$, $\vdash_n^{\alpha+1} \mathtt{tt}$, $\vdash_n^{\alpha+1} \mathtt{ff}$ (any α). For the other constants succ, cond_ι, pcond_ι, $c_=$ and $c_<$ we have rules similar to the Variable Rule, e.g.
  - If $\vdash_n^\alpha M$, then $\vdash_n^{\alpha+1} \mathrm{succ}\, M$.
  - If $\vdash_n^\alpha M$, $\vdash_n^\alpha N$ and $\vdash_n^\alpha L$, then $\vdash_n^{\alpha+1} \mathrm{cond}_\iota MNL$.
- ω^+*-Rule.* If $\alpha = \langle \alpha_\xi \rangle$, $\vdash_n^{\alpha_0} M$ and

  $$\forall \xi \in \Omega_1 \forall m.\, \vdash_0^\xi \mathbf{m} \implies \vdash_n^{\alpha_\xi} RPQ\mathbf{m}\vec{L},$$

  then $\vdash_n^\alpha RPQM\vec{L}$.

- *Cut Rule.* If $\vdash_n^\alpha M\vec{y}$ with $\mathrm{lev}(M) \leq n$ and $\vdash_n^\alpha L_i\vec{y_i}$ for $i = 1, \ldots, m$ with $m \geq 1$, then $\vdash_n^{\alpha+1} ML_1 \ldots L_m$.

- $<_1$*-Rule.* If $\vdash_n^\beta M$ and $\beta <_1 \alpha$, then $\vdash_n^\alpha M$,

where $\beta <_1 \alpha$ means that there is a finite sequence $\beta = \beta_0, \beta_1, \ldots, \beta_n = \alpha$ with $n > 0$ such that for all $i < n$ either $\beta_{i+1} = \beta_i + 1$ or $\beta_{i+1} = \langle \gamma_x \rangle$ and $\beta_i = \gamma_0$ or γ_1, or else $\beta_{i+1} = \langle \gamma_\xi \rangle$ and $\beta_i = \gamma_0$ or γ_1. We also define $\beta \leq_1 \alpha$ to mean $\beta <_1 \alpha$ or $\beta = \alpha$.

More precisely, we first inductively define $\vdash_n^\alpha M$ for $\alpha \in \Omega_1$ by the rules given excluding the ω^+–Rule, and based on this relation we then define $\vdash_n^\alpha M$ for $\alpha \in \Omega_2$ by all the rules given.

3.1. Variable Lemma. *If* $1 \leq_1 \gamma$ *and* $\mathrm{lev}(x) < k$, *then* $\vdash_n^{\gamma \cdot k} x\vec{y}$.

Proof. By induction on $\mathrm{lev}(x)$. By induction hypothesis $\vdash_n^{\gamma\cdot(k-1)} y_i\vec{z_i}$, hence $\vdash_n^{\gamma\cdot(k-1)+1} x\vec{y}$ by the Variable Rule, hence $\vdash_n^{\gamma\cdot k} x\vec{y}$ by the $<_1$–Rule. ∎

3.2. Substitution Lemma. *If* $\vdash_n^\alpha M$ *and* $\vdash_n^\beta N_j\vec{y_j}$ *with* $\mathrm{lev}(N_j) \leq n$ *for* $j = 1, \ldots, m$, *then* $\vdash_n^{\beta+\alpha} M_{\vec{x}}[\vec{N}]$.

Proof. By induction on $\vdash_n^\alpha M$. We write L^* for $L_{\vec{x}}[\vec{N}]$. *Variable Rule*. By induction hypothesis $\vdash_n^{\beta+\alpha} L_i^*\vec{y_i}$, hence $\vdash_n^{\beta+\alpha+1} xL_1^* \ldots L_m^*$ by the Variable Rule. Now if x is one of the variables x_j to be substituted by N_j, we must use the Cut Rule instead of the Variable Rule. This is possible since $\mathrm{lev}(N_j) \leq n$ by hypothesis and $\vdash_n^{\beta+\alpha} N_j\vec{y_j}$ by hypothesis and the $<_1$–Rule. Then (if $m > 0$) the Cut Rule yields $\vdash_n^{\beta+\alpha+1} N_jL_1^* \ldots L_m^*$, as required. In case $m = 0$ there are no L_i's and we have used the Variable Rule to generate $\vdash_n^{\alpha+1} x_j$. But then $\vdash_n^{\beta+\alpha+1} N_j$ holds by hypothesis and the $<_1$–Rule. For all other rules the claim follows easily from the induction hypothesis and the same rule. E.g. for the Constant Rule for cond_ι we have by induction hypothesis $\vdash_n^{\beta+\alpha} M^*$, $\vdash_n^{\beta+\alpha} N^*$ and $\vdash_n^{\beta+\alpha} L^*$. Then an application of the same rule yields $\vdash_n^{\beta+\alpha+1} \mathrm{cond}_\iota M^*N^*L^*$. ∎

3.3. Cut Elimination Lemma. *If* $\vdash_{n+1}^\alpha M$, *then* $\vdash_n^{E\alpha} M$, *where* E *is the exponentiation function* $E\alpha = 2^\alpha$ *defined on* Ω_2 *as follows:*

$$E0 = 1 \qquad E(\alpha+1) = E\alpha + E\alpha \qquad E\langle\alpha_x\rangle = \langle E\alpha_x\rangle \qquad E\langle\alpha_\xi\rangle = \langle E\alpha_\xi\rangle.$$

Proof. By induction on $\vdash_{n+1}^\alpha M$. *Variable Rule*. By induction hypothesis $\vdash_n^{E\alpha} L_i\vec{y_i}$, hence $\vdash_n^{(E\alpha)+1} xL_1 \ldots L_m$ by the Variable Rule, hence $\vdash_n^{E(\alpha+1)} xL_1 \ldots L_m$ by the $<_1$–Rule. *Constant Rules*. Similarly. ω^+*–Rule*. Then $\vdash_{n+1}^\alpha RPQM\vec{L}$ has been inferred from $\vdash_{n+1}^{\alpha_0} M$ with $\alpha = \langle\alpha_\xi\rangle$, and

$$\forall\xi \in \Omega_1 \forall m. \vdash_0^\xi \mathbf{m} \implies \vdash_{n+1}^{\alpha_\xi} RPQ\mathbf{m}\vec{L}.$$

By induction hypothesis $\vdash_n^{E\alpha_0} M$ and

$$\forall \xi \in \Omega_1 \forall m.\ \vdash_0^{\xi} \mathbf{m} \implies \vdash_n^{E\alpha_\xi} RPQ\mathbf{m}\vec{L}.$$

Now $E\alpha_0 = (E\alpha)_0$, and $E\alpha_\xi = (E\alpha)_\xi$. Hence $\vdash_n^{E\alpha} RPQM\vec{L}$ by the ω^+–Rule. *Cut Rule*. By induction hypothesis $\vdash_n^{E\alpha} M\vec{y}$ and $\vdash_n^{E\alpha} L_i\vec{y_i}$. Since $\mathrm{lev}(M) \leq n+1$, we have $\mathrm{lev}(L_i) \leq n$ and hence $\vdash_n^{E\alpha+E\alpha} ML_1 \dots L_m$ by the Substitution Lemma. But $E(\alpha+1) = E\alpha + E\alpha$, so $\vdash_n^{E(\alpha+1)} ML_1 \dots L_m$ immediately. $<_1$*–Rule*. By induction hypothesis $\vdash_n^{E\beta} M$ where $\beta <_1 \alpha$. Since from $\beta <_1 \alpha$ we can easily infer $E\beta <_1 E\alpha$, we get $\vdash_n^{E\alpha} M$ by the $<_1$–Rule. ∎

3.4. Second Collapsing Lemma. *Recall the definition of the collapsing function $D\alpha = D(\alpha, \omega)$ in section 2. Suppose $\alpha \in \Omega_2$ has the property that every big limit $\beta = \langle \beta_\xi \rangle$ occurring in its construction satisfies $\xi \leq_1 \eta \implies \beta_\xi \leq_1 \beta_\eta$. Then we have*

$$\vdash_0^{\alpha} \mathbf{m} \implies \vdash_0^{D\alpha} \mathbf{m}.$$

Proof. By induction on $\vdash_0^{\alpha} \mathbf{m}$. Recall that we identify terms with the same normal form. *Constant Rules*. *Case* 0. Clear, since $\vdash_0^{D1} 0$ by the $<_1$–Rule using $1 <_1 \omega + 1$. *Case* succ. By induction hypothesis $\vdash_0^{D\alpha} \mathbf{m}$, hence $\vdash_0^{D\alpha+1} \mathrm{succ}\ \mathbf{m}$ by the Constant Rule for succ, hence $\vdash_0^{D(\alpha+1)} \mathrm{succ}\ \mathbf{m}$. *Cases* $\mathrm{cond}_{\mathbf{nat}}$, $\mathrm{pcond}_{\mathbf{nat}}$. If $\mathrm{cond}_{\mathbf{nat}} MNL$ or $\mathrm{pcond}_{\mathbf{nat}} MNL$ has a defined value denoted by $\mathbf{m}$, then this is the value of either N or L. Suppose e.g. it is the value of N. Then by induction hypothesis $\vdash_0^{D\alpha} \mathbf{m}$, hence $\vdash_0^{D\alpha+1} \mathbf{m}$ by the same Constant Rule (since we identify terms with the same normal form), hence $\vdash_0^{D(\alpha+1)} \mathbf{m}$. ω^+*–Rule*. Then $\vdash_0^{\alpha} RPQM\vec{L}$ has been inferred from $\vdash_0^{\alpha_0} M$ and

$$\forall \xi \in \Omega_1 \forall n.\ \vdash_0^{\xi} \mathbf{n} \implies \vdash_0^{\alpha_\xi} RPQ\mathbf{n}\vec{L},$$

and $\mathbf{m}$ denotes the value of $RPQM\vec{L}$. Then M must have a defined value, say k. Hence $\mathbf{m}$ also denotes the value of $RPQ\mathbf{k}\vec{L}$. We have to show $\vdash_0^{D\alpha} \mathbf{m}$. By the induction hypothesis we have $\vdash_0^{D\alpha_0} \mathbf{k}$. Then with $n = k$ and $\xi = D\alpha_0$ we get $\vdash_0^{\alpha_{D\alpha_0}} RPQ\mathbf{k}\vec{L}$ from our assumption, so again the induction hypothesis yields $\vdash_0^{D\alpha} \mathbf{m}$ by definition of $D\alpha$. $<_1$*–Rule*. By induction hypothesis $\vdash_0^{D\beta} \mathbf{m}$ where $\beta <_1 \alpha$, hence $\vdash_0^{D\alpha} \mathbf{m}$ by the $<_1$–Rule using the following claim:

$$\beta \leq_1 \alpha \implies D\beta \leq_1 D\alpha.$$

Proof of the claim is by induction over α satisfying the conditions stated above. The zero, successor and small limit cases are immediate. Suppose then that $\alpha = \langle \alpha_\xi \rangle$ where $\alpha_\xi \leq_1 \alpha_\eta$ whenever $\xi \leq_1 \eta$. If $\beta <_1 \alpha$ then $\beta \leq_1 \alpha_i$ by definition where $i = 0$ or 1. But it is straightforward to verify that $i \leq_1 \omega \leq_1 D\gamma$ for every γ. Therefore with $\gamma = \alpha_0$ we obtain $\alpha_i \leq_1 \alpha_{D\alpha_0}$ and hence by the induction hypothesis $D\beta \leq_1 D\alpha_i \leq_1 D\alpha_{D\alpha_0} = D\alpha$. ∎

3.5. First Collapsing Lemma. *Let G_α, $\alpha \in \Omega_1$, be the following slightly modified version of the slow–growing hierarchy:*

$$G_0(n) = 0, \qquad G_{\alpha+1}(n) = G_\alpha(n) + 1, \qquad G_{\langle\alpha_x\rangle}(n) = \max_{i \le n} G_{\alpha_i}(n).$$

Then for all $\alpha \in \Omega_1$ we have

$$\vdash_0^\alpha \mathbf{n} \implies \vdash_0^{G_\alpha(1)} \mathbf{n}.$$

Proof. By induction on $\vdash_0^\alpha \mathbf{n}$ noting that the ω^+–rule cannot be applied. *Constant Rules. Case* 0. $\vdash_0^{G_1(1)} 0$ since $G_1(1) = 1$. *Case* succ. By induction hypothesis $\vdash_0^{G_\alpha(1)} \mathbf{n}$, hence $\vdash_0^{G_{\alpha+1}(1)}$ succ $\mathbf{n}$ by the Constant Rule for succ, since $G_{\alpha+1}(1) = G_\alpha(1) + 1$. *Cases* $\text{cond}_{\mathbf{nat}}$, $\text{pcond}_{\mathbf{nat}}$. $\text{cond}_{\mathbf{nat}} MNL$ or $\text{pcond}_{\mathbf{nat}} MNL$ can only have $\mathbf{n}$ as normal form when either N or L does. Suppose it is N. Then by induction hypothesis $\vdash_0^{G_\alpha(1)} \mathbf{n}$, hence $\vdash_0^{G_{\alpha+1}(1)} \mathbf{n}$. *$<_1$–Rule.* By induction hypothesis $\vdash_0^{G_\beta(1)} \mathbf{n}$ where $\beta <_1 \alpha$. We can then conclude that $\vdash_0^{G_\alpha(1)} \mathbf{n}$ by the following claim:

$$\beta \le_1 \alpha \implies G_\beta(1) \le G_\alpha(1).$$

Proof of claim. The cases $\alpha = 0$ and $\alpha = \alpha' + 1$ are immediate. Suppose that $\alpha = \langle\alpha_x\rangle$. Then if $\beta <_1 \alpha$ we have $\beta \le_1 \alpha_i$ where $i = 0$ or $i = 1$. The induction hypothesis then gives $G_\beta(1) \le G_{\alpha_i}(1)$ and hence $G_\beta(1) \le G_\alpha(1)$ by the revised limit clause in the definition of G_α. ∎

3.6. Value Lemma. *If $\vdash_0^k \mathbf{n}$, then $n < k$.*

Proof. By induction on $\vdash_0^k \mathbf{n}$. *Constant Rules. Case* 0. Obvious. *Case* succ. By induction hypothesis $n < k$, hence $n + 1 < k + 1$. *Cases* $\text{cond}_{\mathbf{nat}}$, $\text{pcond}_{\mathbf{nat}}$. $\text{cond}_{\mathbf{nat}} MNL$ or $\text{pcond}_{\mathbf{nat}} MNL$ can only have $\mathbf{n}$ as normal form when either N or L does. Suppose it is N. Then by induction hypothesis $n < k$, hence $n < k + 1$. *$<_1$–Rule.* Note first that from $\beta <_1 k$ we can conclude that β is a numeral l with $l < k$. By induction hypothesis $n < l$, hence $n < k$.o ∎

We now construct an "initial" generation tree for any term M. The main point here is that by using the Cut Rule we can "shortcut" an enormous generation tree. Consider for example an application term $(\lambda x M)N$ where N may be complex and M may contain many occurrences of x. The shortcut is achieved by considering instead of $(\lambda x M)N$, which may have a complex normal form, the two terms $(\lambda x M)y$ (with a variable y) and N separately.

3.7. Embedding Lemma. *Assume that all subterms of M have levels $\le n$. Then we can find k such that $\vdash_n^{\Omega \cdot k} M\vec{y}$, where $\Omega := \langle\xi\rangle_{\xi \in \Omega_1}$.*

Proof. By induction on M. *Case* x. The claim follows from the Variable Lemma with $k := \mathrm{lev}(x) + 1$. *Cases* $\perp^{\mathbf{nat}}$, 0, $\perp^{\mathbf{boole}}$, $\mathbb{tt}$, $\mathbb{ff}$. Obvious. *Case* succ. By the Variable Rule $\vdash^1_n y$, hence $\vdash^\Omega_n y$ by the $<_1$–Rule, hence $\vdash^{\Omega+1}_n \mathrm{succ}\, y$ by the Constant Rule for succ, hence $\vdash^{\Omega \cdot 2}_n \mathrm{succ}\, y$ again by the $<_1$–Rule. *Cases* cond_ι, pcond_ι, $c_=$, $c_<$. As for Case succ. *Case* R. Consider $Rywx\vec{y}$. We want to apply the ω^+–Rule to get $\vdash^{\Omega\cdot(k+1)}_n Rywx\vec{y}$ with $k = \max(\mathrm{lev}(y), 1)$. Since $(\Omega \cdot (k+1))_0 = \Omega \cdot k$ and $\vdash^{\Omega\cdot k}_0 x$ by the $<_1$–Rule, we only have to show

$$\forall \xi \in \Omega_1 \forall m.\, \vdash^\xi_0 \mathbf{m} \implies \vdash^{\Omega\cdot k+\xi}_n Rywm\vec{y}.$$

This is done by induction on $\vdash^\xi_0 \mathbf{m}$. Note that $\vdash^\xi_0 \mathbf{m}$ can only be derived by a Constant Rule or the $<_1$–Rule (the ω^+–Rule is excluded, since $\xi \in \Omega_1$). *Constant Rules*. *Case* 0. Since $Ryw0\vec{y}$ is identified with $y\vec{y}$ it suffices to show $\vdash^{\Omega\cdot k+1}_n y\vec{y}$. Since $\mathrm{lev}(y_i) < k$, this follows from the Variable Lemma and the Variable Rule. *Case* succ. By induction hypothesis $\vdash^{\Omega\cdot k+\xi}_n Ryw\mathbf{m}\vec{y}$. Since $Ryw(\mathrm{succ}\ \mathbf{m})\vec{y}$ is identified with $w\mathbf{m}(Ryw\mathbf{m})\vec{y}$ it suffices to show $\vdash^{\Omega\cdot k+\xi+1}_n w\mathbf{m}(Ryw\mathbf{m})\vec{y}$. This follows from the Variable Rule, since from $\vdash^\xi_0 \mathbf{m}$ we get $\vdash^{\Omega\cdot k+\xi}_0 \mathbf{m}$ by the Substitution Lemma. *Cases* $\mathrm{cond}_{\mathbf{nat}}$, $\mathrm{pcond}_{\mathbf{nat}}$. $\mathrm{cond}_{\mathbf{nat}}MNL$ or $\mathrm{pcond}_{\mathbf{nat}}MNL$ can only have $\mathbf{m}$ as normal form when either N or L does. Suppose it is N. Then by induction hypothesis $\vdash^{\Omega\cdot k+\xi}_n Ryw\mathbf{m}\vec{y}$, hence $\vdash^{\Omega\cdot k+\xi+1}_n Ryw\mathbf{m}\vec{y}$ by the $<_1$–Rule. *$<_1$–Rule*. The claim follows from the induction hypothesis and the $<_1$–Rule. *Case* $\lambda x M$. By induction hypothesis $\vdash^{\Omega\cdot k}_n M\vec{y}$, which says $\vdash^{\Omega\cdot k}_n (\lambda x M)x\vec{y}$, since both terms have the same normal form and hence are identified. *Case* MN. By induction hypothesis $\vdash^{\Omega\cdot k}_n My\vec{y}$ and $\vdash^{\Omega\cdot k}_n N\vec{x}$. Also $\vdash^{\Omega\cdot k}_n y_i\vec{y}_i$. Note that by the $<_1$–Rule we can assume that we have the same k in all cases. Then $\vdash^{\Omega\cdot k+1}_n MN\vec{y}$ by the Cut Rule, hence $\vdash^{\Omega\cdot(k+1)}_n MN\vec{y}$ by the $<_1$–Rule. ∎

Now we obtain the desired estimate in terms of the slow–growing subrecursive hierarchy.

3.8. Theorem. *Let M be a closed term of type* $\mathbf{nat} \to \mathbf{nat}$. *Then there is a k such that for all $n \geq 1$ the value of $M\mathbf{n}$ is either undefined or $\leq G_\alpha(n)$, with $\alpha := D(E^k(\Omega \cdot k) \cdot (1 + \omega))$.*

Proof. Consider Mx. By the Embedding Lemma we find a k such that $\vdash^{\Omega\cdot k}_k Mx$. Hence $\vdash^\gamma_0 Mx$ with $\gamma := E^k(\Omega \cdot k)$ by the Cut Elimination Lemma. Note that $\vdash^{\gamma\cdot n}_0 \mathbf{n}$; this follows from

$$\gamma = E^k(\Omega \cdot k) >_1 E^k(\Omega \cdot (k-1) + 1) \geq_1 E^k(1) \geq_1 2$$

by induction an n: For $n = 1$ we obtain $\vdash^\gamma_0 1$ from $\vdash^2_0 1$ by the $<_1$–Rule, and from $\vdash^{\gamma\cdot n}_0 \mathbf{n}$ we first obtain $\vdash^{\gamma\cdot n+1}_0 (\mathbf{n+1})$ and then using $1 <_1 \gamma$ and the $<_1$–Rule $\vdash^{\gamma\cdot(n+1)}_0 (\mathbf{n+1})$.

So we have $\vdash_0^{\gamma\cdot(n+1)} M\mathbf{n}$ by the Substitution Lemma. Let $\mathbf{m}$ denote the numerical value of $M\mathbf{n}$ (assuming it is defined). Then with $\alpha := D(\gamma\cdot(1+\omega))$ we have $\alpha_n = D(\gamma\cdot(n+1))$ and hence $\vdash_0^{\alpha_n} \mathbf{m}$ by the Second Collapsing Lemma, since it is easy to check that Ω, $\Omega\cdot k$ and $E^k(\Omega\cdot k)$ all satisfy the required condition (the only important thing to notice is that E preserves the $\leq_1$–ordering). Hence $\vdash_0^{G_{\alpha_n}(1)} \mathbf{m}$ by the First Collapsing Lemma and $m < G_{\alpha_n}(1)$ by the Value Lemma. But the G_α–functions are increasing and so

$$m < G_{\alpha_n}(1) \leq G_{\alpha_n}(n) \leq G_\alpha(n).$$

∎

Proof. Consider $M(\text{succ } x)$. By the Embedding Lemma we find a k such that $\vdash_k^{\Omega\cdot k} M(\text{succ } x)$. Hence $\vdash_0^{\gamma} M(\text{succ } x)$ with $\gamma := E^k(\Omega\cdot k)$ by the Cut Elimination Lemma. Since $\vdash_0^{\gamma\cdot n} (\mathbf{n-1})$ (by the Constant Rules and the $<_1$–Rule) we get $\vdash_0^{\gamma\cdot(n+1)} M\mathbf{n}$ by the Substitution Lemma. Let $\mathbf{m}$ denote the numerical value of $M\mathbf{n}$ (assuming it is defined). Then with $\alpha := D(\gamma\cdot(1+\omega))$ we have $\alpha_n = D(\gamma\cdot(n+1))$ and hence $\vdash_0^{\alpha_n} \mathbf{m}$ by the Second Collapsing Lemma, since it is easy to check that Ω, $\Omega\cdot k$ and $E^k(\Omega\cdot k)$ all satisfy the required condition (the only important thing to notice is that E preserves the $\leq_1$–ordering). Hence $\vdash_0^{G_{\alpha_n}(1)}\mathbf{m}$ by the First Collapsing Lemma and $m < G_{\alpha_n}(1)$ by the Value Lemma. But the G_α–functions are increasing and so

$$m < G_{\alpha_n}(1) \leq G_{\alpha_n}(n) \leq G_\alpha(n).$$

∎

Since for each term M of type **nat** there is another M' whose value bounds the lengths of reduction sequences for M, it follows that to each term M of type **nat** $\rightarrow$ **nat** there is associated a tree ordinal α below the Bachmann–Howard ordinal $\sup_k D(E^k(\Omega\cdot k)\cdot(1+\omega))$ such that the slow–growing function G_α bounds, for each input n, the length of any reduction sequence starting with $M\mathbf{n}$. In the terminology of section 2 we thus obtain

$$||\text{PCF}_<|| \leq \text{ Bachmann–Howard ordinal.}$$

This is in fact a tight bound. It is well known that all of the slow–growing functions below the Bachmann–Howard ordinal can be defined in $\text{PCF}_<$.

Results of [8] then show that α acts as a termination ordering over which the numerical function defined by the program M may be evaluated by term-rewriting. Known results about the connection between "slow" and "fast" growing functions then give immediate access to various proof–theoretic characterizations of functions definable by $\text{PCF}_\prec$ programs, for example that the functions definable in $\text{PCF}_<$ are exactly the provably recursive functions of arithmetic.

REFERENCES

[1] S. Abramsky. Semantics of PCF. Unpublished Notes, 1989.

[2] T. Arai. A slow growing analogue to Buchholz' proof. *Annals of Pure and Applied Logic*, 54:101–120, 1991.

[3] W. Buchholz. A new system of proof–theoretic ordinal functions. *Annals of Pure and Applied Logic*, 32(3):195–207, 1986.

[4] W. Buchholz. An independence result for Π^1_1–CA+BI. *Annals of Pure and Applied Logic*, 33(2):131–155, 1987.

[5] W. Buchholz, S. Feferman, W. Pohlers, and W. Sieg. *Iterated Inductive Definitions and Subsystems of Analysis: Recent Proof–Theoretical Studies*, volume 897 of *Lecture Notes in Mathematics*. Springer, Berlin, 1981.

[6] W. Dosch. Reduction relations in strict applicative languages. In *ISTCS '92*, pages 55–66. Springer LNCS 601, 1992.

[7] M. V. Fairtlough and S. S. Wainer. Ordinal complexity of recursive definitions. *Information and Computation*, 99:123–153, 1992.

[8] W. G. Handley and S. S. Wainer. Equational derivation vs. computation. To appear: Annals of Pure and Applied Logic.

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

[10] H. Schwichtenberg. Primitive recursion on the partial continuous functionals. In M. Broy, editor, *Informatik und Mathematik*, pages 251–269. Springer, Berlin, 1991.

[11] H. Schwichtenberg. Proofs as programs. In P. Aczel, H. Simmons, and S. S. Wainer, editors, *Proof Theory. A selection of papers from the Leeds Proof Theory Programme 1990*, pages 81–113. Cambridge University Press, 1992.

[12] H. Schwichtenberg. Type theory. Course notes, Universität München, 1992.

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

[14] A. S. Troelstra and D. van Dalen. *Constructivism in Mathematics. An Introduction*, volume 121, 123 of *Studies in Logic and the Foundations of Mathematics*. North–Holland, Amsterdam, 1988.

[15] S. S. Wainer. Slow growing versus fast growing. *Journal of Symbolic Logic*, 54(2):608–614, June 1989.

Helmut Schwichtenberg
Fakultät für Mathematik
Universität München
Theresienstr 39
D-8000 München 2
Germany
schwicht@rz.mathematik.uni-muenchen.de

Stanley S. Wainer
School of Mathematics
University of Leeds
Leeds LS2 9JT
England
pmt6ssw@sun.leeds.ac.uk