ELSEVIER Annals of Pure and Applied Logic 94 (1998) 201–222

ANNALS OF PURE AND APPLIED LOGIC

Finite notations for infinite terms

Helmut Schwichtenberg
Department of Mathematics, University of Munich, Munich, Germany

Abstract

Buchholz (1991) presented a method to build notation systems for infinite sequent-style derivations, analogous to well-known systems of notation for ordinals. The essential feature is that from a notation one can read off by a *primitive* (not ε_0-) recursive function its nth predecessor and, e.g. the last rule applied. Here we extend the method to the more general setting of infinite (typed) terms, in order to make it applicable in other proof-theoretic contexts as well as in recursion theory. As examples, we use the method to (1) give a new proof of a well-known trade-off theorem (Schwichtenberg, 1975), which says that detours through higher types can be eliminated by the use of transfinite recursion along higher ordinals, and (2) construct a continuous normalization operator with an explicit modulus of continuity. © 1998 Elsevier Science B.V. All rights reserved.

0. Introduction

It is well known that in order to study primitive recursion in higher types it is useful to unfold the primitive recursion operators into infinite terms. A similar phenomenon occurs in proof theory, where one expands induction axioms. For applications it then is often necessary to code these infinite objects by natural numbers. A standard method to design such a coding is to proceed as in Kleene's system $\mathcal{O}$ of ordinal notations; cf. [6] for a recursion theoretic and [7] for a proof theoretic application of this method. However, working with such codes is not easy. For instance, to prove that the standard operation reducing the cut rank by one can be represented by a primitive recursive operation on codes requires some careful applications of Kleene's recursion theorem for primitive recursive functions.

Now, Buchholz in [1] proposed to use a method familiar from systems of ordinal notations to construct codings of infinite derivations. 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 derviation 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 by Schütte, Tait and Mints [4]. 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 $\mathrm{o}(\cdot)$ such that $h[n]$ denotes the nth premise of the infinite derivation $[\![h]\!]$ denoted by h, and $\mathrm{o}(h)$ 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 [4].

The aim of the present paper is to adapt Buchholz' approach to the more general and more flexible setting of infinite (typed) terms. At first sight it might seem that this should be a routine matter; however, it turned out that this is not so.

One benefit of having the method available for inifinite terms is that applications in recursion theory as well as in proof theory become possible. As an example we prove that in $\mathrm{P{\scriptstyle CF}}_\alpha$, a partial verison of Gödel's T with transfinitely bounded fixed point operators (cf. [10, 9]), one can eliminate detours through higher types by raising the ordinals of the transfinite recursions involved. The first result of this kind is due to Kreisel [3]; here we discuss a generalization that first appeared in [6]. There the standard method to code infinite objects – as described above – had been applied. In the present paper we give a new proof of this result, as an example of the machinery developed. (Another proof of the trade-off theorem can be obtained from a recent rather elegant analysis of Gödel's T by Weiermann [12].) As a second example, we consider possibly non-well-founded infinite terms. It is well known that similar to Mints' continuous cut-elimination operator [4] one can define a continuous normalization operator; in [8] a somewhat intuitive definition of such an operator was given. Here it is shown that Buchholz' formally quite elegant treatment of continuous cut elimination in [1] can be adapted to non-well-founded terms as well, yielding an explicit expression for the modulus of continuity.

1. Infinite terms with the repetition rule

We describe a system of infinite (typed) terms as in Tait [11], and also the well-known reduction operator $\mathrm{R{\scriptstyle ED}}$ reducing the cut-rank by at least one.

The set of (simple) types is defined inductively from the ground type ι of natural numbers by $[\rho \to \sigma]$. The *level* of a type is defined by $\mathrm{lev}(\iota) := 0$, $\mathrm{lev}(\rho \to \sigma) := \max(\mathrm{lev}(\rho) + 1, \mathrm{lev}(\sigma))$. Infinite (typed) terms are defined inductively by the clauses v_i^ρ, $c\vec{t}$, $\lambda v_i^\rho t$, ts, $\langle t_i^\iota \rangle_{i\in\mathbb{N}}$, $\textsc{Rep}(t)$. We write t, s, r for infinite terms, and x, y, z for variables. $\mathrm{Tp}(t)$ denotes the type of t. We assume that finitely many constants c denoting primitive recursive functions are given, where each such function expects $n \geqslant 0$ arguments of ground type ι and yields a value of ground type ι; in addition, for every $k \in \mathbb{N}^\perp := \{\perp, 0, 1, 2, \ldots\}$ (where $\perp$ is viewed as the object 'undefined') we have a constant $\underline{k}$. Let $\textsc{Terms}$ denote the set of all such infinite terms. The *level* of a term is defined to be the level of its type.

For every inifinite term t we define an ordinal $|t|$ – the *depth* of t – by

$$|x| := 0,$$

$$|ct_1 \cdots t_n| := \max(|t_1|, \ldots, |t_n|),$$

$$|\lambda x t| := |t| + 1,$$

$$|ts| := \max(|t|, |s|) + 1,$$

$$|\langle t_i \rangle_{i\in\mathbb{N}}| := \sup_i (|t_i| + 1),$$

$$|\textsc{Rep}(t)| := |t| + 1.$$

1.1. *Reducing the rank of a term.* The *rank* of an infinite term is 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 $(\textsc{Rep}(t))s$. So the rank function $\mathrm{Rk} : \textsc{Terms} \to \mathbb{N} \cup \{\infty\}$ is defined by

$$\mathrm{Rk} := 0,$$

$$\mathrm{Rk}(ct_1 \cdots t_n) := \max(\mathrm{Rk}(t_1), \ldots, \mathrm{Rk}(t_n)),$$

$$\mathrm{Rk}(\lambda x t) := \mathrm{Rk}(t),$$

$$\mathrm{Rk}(ts) := \begin{cases} \max(\mathrm{lev}(\rho), \mathrm{Rk}(t), \mathrm{Rk}(s)) & \text{if } \rho = \mathrm{Tp}(t) \text{ and} \\ & t = \lambda x t_0 \text{ or } t = \textsc{Rep}(t_0), \\ \max(\mathrm{Rk}(t), \mathrm{Rk}(s)) & \text{otherwise,} \end{cases}$$

$$\mathrm{Rk}(\langle t_i \rangle_{i\in\mathbb{N}}) := \sup_i \mathrm{Rk}(t_i),$$

$$\mathrm{Rk}(\textsc{Rep}(t)) := \mathrm{Rk}(t).$$

Also, for every term t we define the set $\mathrm{FV}(t)$ of its free variables as usual. It suffices to restrict attention to terms t with $\mathrm{Rk}(t) \neq \infty$ and $\mathrm{FV}(t)$ finite. Let $\textsc{Terms}^\circ$ denote the set of all such terms, and t, s, r from now on range over $\textsc{Terms}^\circ$.

Note that it is not necessary to also take into account redexes of the form $\langle t_i \rangle_{i\in\mathbb{N}}\underline{k}$. The reason is that they do not cause any detours through higher-type levels for the valuation functionals in Section 4.4.

In order to define a reduction function RED reducing the rank we first have to define a substitution function and then a function BETA corresponding to β-conversion.

For every n-tuple $x_1,\dots,x_n =: \vec{x}$ of distinct variables, every n-tuple $s_1,\dots,s_n =: \vec{s}$ of terms with $\mathrm{Tp}(s_i)=\mathrm{Tp}(x_i)$ and every term $t \in \mathrm{TERMS}^\circ$ we define the term $\mathrm{SUB}_{\vec{x}}(t,\vec{s})$ as usual; in case λxt we (for simplicity) rename the bound variable even when this is not necessary.

Lemma. $\mathrm{SUB}_{\vec{x}}$ *has the following properties.*

(i) $\mathrm{Tp}(\mathrm{SUB}_{\vec{x}}(t,\vec{s}))=\mathrm{Tp}(t)$.

(ii) $\mathrm{FV}(\mathrm{SUB}_{\vec{x}}(t,\vec{s})) = \bigcup_{z\in \mathrm{FV}(t)} \begin{cases} \mathrm{FV}(s_i) & \text{if } z=x_i,\\ \{z\} & \text{otherwise,}\end{cases}$

(iii) $\mathrm{Rk}(\mathrm{SUB}_{\vec{x}}(t,\vec{s})) \leqslant \max(\mathrm{lev}(\vec{\rho}^\circ), \mathrm{Rk}(\vec{s}), \mathrm{Rk}(t))$, *where* $\vec{\rho}^\circ$ *consists of those* ρ_i *from* $\vec{\rho} := \mathrm{Tp}(\vec{x})$ *with* s_i *an abstraction or a repetition,*

(iv) $|\mathrm{SUB}_{\vec{x}}(t,\vec{s})| \leqslant (\max|\vec{s}|) + |t|$.

So in particular $\mathrm{SUB}_{\vec{x}}(t,\vec{s}) \in \mathrm{TERMS}^\circ$.

Proof. (i)–(iv) are each proved easily by induction on t. In (iii), the only case when a new redex is produced is an occurrence of a subterm $x_i s$ in t with s_i either an abstraction or a repetition. But this is taken care of in the definition of Rk. □

Now we can define $\mathrm{BETA}(t,s)$ for terms t,s such that $\mathrm{Tp}(t)=[\mathrm{Tp}(s)\to\rho]$, as follows:

$$\mathrm{BETA}(t,s) := \begin{cases} \mathrm{REP}(\mathrm{SUB}_x(t_0,s)) & \text{if } t=\lambda x t_0,\\ \mathrm{REP}(\mathrm{BETA}(t_0,s)) & \text{if } t=\mathrm{REP}(t_0),\\ ts & \text{otherwise.}\end{cases}$$

The reason for the insertion of REP in the first case will be apparent later, when we define $\mathrm{rl}(\beta(a,b))$.

Lemma. *Let* $\mathrm{Tp}(t)=[\mathrm{Tp}(s)\to\rho]$. *Then*

(i) $\mathrm{Tp}(\mathrm{BETA}(t,s))=\rho$,

(ii) $\mathrm{FV}(\mathrm{BETA}(t,s)) \subseteq \mathrm{FV}(t)\cup\mathrm{FV}(s)$,

(iii) $\mathrm{Rk}(\mathrm{BETA}(t,s)) \leqslant \max(\mathrm{lev}(s),\mathrm{Rk}(t),\mathrm{Rk}(s))$,

(iv) $|\mathrm{BETA}(t,s)| \leqslant |s|+|t|+1$.

Proof. (i)–(iii) can be proved easily by induction on t, using the properties of the SUB function. For (iv) we also use induction on t:

$$|\mathrm{BETA}(\lambda x t_0,s) \leqslant |s|+|t_0|+1 = |s|+|t|,$$
$$|\mathrm{BETA}(\mathrm{REP}(t_0,s)| = |\mathrm{BETA}((t_0,s)| + 1 \leqslant_{\mathrm{IH}} |s|+|t_0|+2 = |s| + |\mathrm{REP}(t_0)|+1$$

and otherwise $|\mathrm{BETA}(t,s)| = |ts| = \max(|t|,|s|)+1 \leqslant |s|+|t|+1$. □

Note that $|\textsc{Beta}(t,s)| \leqslant |s| + |t|$ does not hold generally; a counterexample is $|\textsc{Beta}(\langle f^i x\rangle_{i\in\mathbb{N}}, y)| = |\langle f^i x\rangle_{i\in N} y| = \omega + 1$, but $|y| + |\langle f^i x\rangle_{i\in\mathbb{N}}| = |y| + \omega = \omega$.

Now, we can define a reduction function $\textsc{Red}: \textsc{Terms}^\circ \to \textsc{Terms}$ reducing the rank of a term by at least one:

$$\begin{aligned}
&\textsc{Red}(x) := x,\\
&\textsc{Red}(ct_1 \cdots t_n) := c\,\textsc{Red}(t_1)\cdots\textsc{Red}(t_n),\\
&\textsc{Red}(\lambda x t) := \lambda x\,\textsc{Red}(t)\\
&\textsc{Red}(ts) := \begin{cases} \textsc{Rep}(\textsc{Beta}(\textsc{Red}(t), \textsc{Red}(s))) & \text{if } ts \text{ is a critical application,}\\ \textsc{Red}(t)\textsc{Red}(s) & \text{otherwise,}\end{cases}\\
&\textsc{Red}(\langle t_i\rangle_{i\in\mathbb{N}}) := \langle \textsc{Red}(t_i)\rangle_{i\in\mathbb{N}},\\
&\textsc{Red}(\textsc{Rep}(t)) := \textsc{Rep}(\textsc{Red}(t)).
\end{aligned}$$

Here an application term is called a *critical application* if it is of the from $(\lambda x t)\vec{t}$ or $(\textsc{Rep}(t))\vec{t}$; otherwise it is called a *simple application.*

The reason for the insertion of $\textsc{Rep}$ in the first part of case ts will again be apparent later (cf. the definition of $\mathrm{rl}(\mathrm{red}(a))$ below).

Lemma. *For every* $t \in \textsc{Terms}^\circ$ *we have*
(i) $\mathrm{FV}(\textsc{Red}(t)) \subseteq \mathrm{FV}(t)$,
(ii) $\mathrm{Rk}(\textsc{Red}(t)) < \mathrm{Rk}(t)$ *if* $0 < \mathrm{Rk}(t)$, *and* $\mathrm{Rk}(\textsc{Red}(t)) = 0$ *if* $\mathrm{Rk}(t) = 0$.
(iii) $|\textsc{Red}(t)| \leqslant 3^{|t|}$.
So, in particular, $\textsc{Red}(t) \in \textsc{Terms}^\circ$.

Proof. (i) follows from $\mathrm{FV}(\textsc{Beta}(t,s)) \subseteq \mathrm{FV}(t) \cup \mathrm{FV}(s)$.

(ii) It clearly suffices to consider an application term. Now, a simple application is of the form $xt_1 \cdots t_n$ or $\langle t_i\rangle_{i\in\mathbb{N}} s$, and by definition $\textsc{Red}(xt_1\cdots t_n) = x\textsc{Red}(t_1)\cdots\textsc{Red}(t_n)$ and $\textsc{Red}(\langle t_i\rangle_{i\in\mathbb{N}} s) = \langle\textsc{Red}(t_i)\rangle_{i\in\mathbb{N}}\textsc{Red}(s)$. So we only need to consider the case of a critical application ts. But then $\mathrm{Rk}(ts) > \mathrm{lev}(s)$ and we obtain

$$\begin{aligned}
\mathrm{Rk}(\textsc{Red}(ts)) &= \mathrm{Rk}(\textsc{Rep}(\textsc{Beta}(\textsc{Red}(t), \textsc{Red}(s))))\\
&= \mathrm{Rk}(\textsc{Beta}(\textsc{Red}(t), \textsc{Red}(s)))\\
&\leqslant \max(\mathrm{lev}(s), \mathrm{Rk}(\textsc{Red}(t)), \mathrm{Rk}(\textsc{Red}(s)))\\
&< \mathrm{Rk}(ts) \quad \text{by IH for } t, s.
\end{aligned}$$

(iii) We restrict ourselves to the case of a critical application. *Subcase* $\textsc{Red}(t) = \lambda x t_0$:

$$\begin{aligned}
|\textsc{Red}(ts)| &= |\textsc{Rep}(\textsc{Beta}(\lambda x t_0, \textsc{Red}(s)))|\\
&= |\textsc{Rep}^2(\textsc{Sub}_x(t_0, \textsc{Red}(s)))|\\
&\leqslant |\textsc{Red}(s)| + |t_0| + 2\\
&= |\textsc{Red}(s)| + |\textsc{Red}(t)| + 1\\
&\leqslant 3^{|s|} + 3^{|t|} + 1 \quad \text{by IH}\\
&\leqslant 3^{\max(|t|,|s|)+1}\\
&= 3^{|ts|}.
\end{aligned}$$

Subcase $\textsc{Red}(t) = \textsc{Rep}(t_0)$:

$$\begin{aligned}
|\textsc{Red}(ts)| &= |\textsc{Rep}(\textsc{Beta}(\textsc{Rep}(t_0), \textsc{Red}(s)))| \\
&= |\textsc{Rep}^2(\textsc{Beta}(t_0, \textsc{Red}(s)))| \\
&\leqslant |\textsc{Red}(s)| + |t_0| + 3 \\
&= |\textsc{Red}(s)| + |\textsc{Red}(t)| + 2 \\
&\leqslant 3^{|s|} + 3^{|t|} + 2 \quad \text{by IH} \\
&\leqslant 3^{\max(|t|,|s|)+1} \quad \text{since } 1 \leqslant |t| \\
&= 3^{|ts|}.
\end{aligned}$$

Note that $1 \leqslant |t|$ must hold since otherwise t would be a variable or a constant, hence $\textsc{Red}(t) = t$ and therefore $\textsc{Red}(t) \neq \textsc{Rep}(t_0)$. *Subcase* $\textsc{Red}(t)$ neither an abstraction nor a repetition.

$$\begin{aligned}
|\textsc{Red}(ts)| &= |\textsc{Rep}(\textsc{Red}(t)\textsc{Red}(s))| \\
&= |\textsc{Red}(t)\textsc{Red}(s)| + 1 \\
&= \max(|\textsc{Red}(t)|, |\textsc{Red}(s)|) + 2 \\
&\leqslant \max(3^{|t|}, 3^{|s|}) + 2 \quad \text{by IH} \\
&\leqslant 3^{\max(|t|,|s|)+1} \\
&= 3^{|ts|}. \qquad \square
\end{aligned}$$

So – following Tait [11] – we have shown that every $t \in \textsc{Terms}^\circ$, i.e. every infinite term with finite rank $m := \mathrm{Rk}(t)$ and at most finitely many free variables, can be reduced to normal form, i.e. to a term $\textsc{Red}^m(t) \in \textsc{Terms}^\circ$ of rank zero.

2. 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 nth predecessor of t (as $a[n]$). We will also introduce a 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 (and have to rename bound variables).

We define a set $\textsc{Rules}$ of formal symbols. Also, for every element R of $\textsc{Rules}$ we define its arity $\mathrm{arity}(R) \in \mathbb{N} \cup \{\infty\}$.

(i) For every variable x let $\textsc{Var}_x \in \textsc{Rules}$, and $\mathrm{arity}(\textsc{Var}_x) := 0$;
(ii) for every constant c denoting an n-ary function let $\textsc{Const}_c \in \textsc{Rules}$ and arity $(\textsc{Const}_c) := n$;
(iii) for every variable x let $\textsc{Abs}_x \in \textsc{Rules}$, and $\mathrm{arity}(\textsc{Abs}_x) := 1$;
(iv) for every variable x let $\textsc{Ap}_x \in \textsc{Rules}$, $\textsc{SeqAp} \in \textsc{Rules}$ (for sequence application), $\textsc{CAp} \in \textsc{Rules}$ (for critical application); moreover $\mathrm{arity}(\textsc{Ap}_x) := \mathrm{arity}(\textsc{SeqAp}) := \mathrm{arity}(\textsc{CAp}) := 2$;
(v) $\textsc{Seq} \in \textsc{Rules}$, and $\mathrm{arity}(\textsc{Seq}) := \infty$;
(vi) $\textsc{Rep} \in \textsc{Rules}$, and $\mathrm{arity}(\textsc{Rep}) := 1$.

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

$\mathrm{tp} : T \to$ (codes of) types,
$\mathrm{fv} : T \to$ (codes of) finite sets of variables,
$\mathrm{rl} : T \to$ (codes of) symbols from $\textsc{Rules}$,
$\cdot[\cdot] : T \times \mathbb{N} \to T \cup \{0\}$ (we assume $0 \notin T$)

such that

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

We may use any (obvious) coding device here, with the sole requirement that the natural coding and decoding functions are primitive recursive. $\mathrm{fv}(a)$ is meant to give for every notation a of a term t a finite set of variables containing all those free in t. Let $0[n] := 0$.

A notation system $\mathscr{T} = (T, \mathrm{tp}, \mathrm{fv}, \mathrm{rl}, .[.])$ is *correct* if for every notation a and its predecessors the assigned types and sets of free variables satisfy some obvious conditions. Specifically, we require that for all $a \in T$ and all $k \in \mathbb{N}$ we have

(i) If $\mathrm{rl}(a) = \textsc{Var}_x$, then $\mathrm{tp}(a) = \mathrm{Tp}(x)$ and $x \in \mathrm{fv}(a)$.
(ii) If $\mathrm{rl}(a) = \textsc{Const}_c$, then $\mathrm{tp}(a) = \iota$, $\mathrm{tp}(a[i]) = \iota$ and $\mathrm{fv}(a[i]) \subseteq \mathrm{fv}(a)$, for $i < \mathrm{arity}(\textsc{Const}_c)$.
(iii) If $\mathrm{rl}(a) = \textsc{Abs}_x$, then $\mathrm{tp}(a) = [\mathrm{Tp}(x) \to \mathrm{tp}(a[0])]$ and $\mathrm{fv}(a[0]) \setminus \{x\} \subseteq \mathrm{fv}(a)$.
(iv) If $\mathrm{rl}(a) \in \{\textsc{Ap}_x, \textsc{SeqAp}, \textsc{CAp}\}$, then $\mathrm{tp}(a[0]) = [\mathrm{tp}(a[1]) \to \mathrm{tp}(a)]$ and also $\mathrm{fv}(a[0]) \cup \mathrm{fv}(a[1]) \subseteq \mathrm{fv}(a)$. Moreover, if $\mathrm{rl}(a) = \textsc{Ap}_x$, then we have $\mathrm{rl}(a[0]) \in \{\textsc{Var}_x, \textsc{Ap}_x\}$; if $\mathrm{rl}(a) = \textsc{SeqAp}$, then $\mathrm{rl}(a[0]) \in \{\textsc{Seq}, \textsc{SeqAp}\}$; if $\mathrm{rl}(a) = \textsc{CAp}$, then $\mathrm{rl}(a[0]) = \textsc{Abs}_x$ for some x or else $\mathrm{rl}(a[0]) = \textsc{CAp}$.
(v) If $\mathrm{rl}(a) = \textsc{Seq}$, then $\mathrm{tp}(a[k]) = \iota$, $\mathrm{tp}(a) = [\iota \to \iota]$ and $\mathrm{fv}(a[k]) \subseteq \mathrm{fv}(a)$.
(vi) If $\mathrm{rl}(a) = \textsc{Rep}$, then $\mathrm{tp}(a) = \mathrm{tp}(a[0])$ and $\mathrm{fv}(a[0]) \subseteq \mathrm{fv}(a)$.

$\mathscr{T}$ is *well founded* if there is no infinite sequence $(n_i)_{i \in \mathbb{N}}$ such that $a[n_0] \dots [n_{k-1}] \in T$ for all $k \in \mathbb{N}$.

Let $\prec$ be a well ordering of $\mathbb{N}$. A *$\prec$-norm* for a notation system $\mathscr{T}$ is a function $o : T \to \mathbb{N}$ such that for all $a \in T$ and all $n < \mathrm{arity}(\mathrm{rl}(a))$ we have $o(a[n]) \prec o(a)$. A notation system $\mathscr{T}$ with a $\prec$-norm clearly is well founded.

Let $\mathscr{T}$ be a correct notation system with $\prec$-norm $o : T \to \mathbb{N}$. For every $a \in T$ we define the term t_a denoted by a, via transfinite induction on $o(a)$:

$$t_a := \begin{cases} x & \text{if } \mathrm{rl}(a) = \textsc{Var}_x, \\ c t_{a[0]} \dots t_{a[n-1]} & \text{if } \mathrm{rl}(a) = \textsc{Const}_c, \\ \lambda x\, t_{a[0]} & \text{if } \mathrm{rl}(a) = \textsc{Abs}_x, \\ t_{a[0]} t_{a[1]} & \text{if } \mathrm{rl}(a) \in \{\textsc{Ap}_x, \textsc{SeqAp}, \textsc{CAp}\}, \\ \langle t_{a[i]} \rangle_{i \in \mathbb{N}} & \text{if } \mathrm{rl}(a) = \textsc{Seq}, \\ \textsc{Rep}(t_{a[0]}) & \text{if } \mathrm{rl}(a) = \textsc{Rep}. \end{cases}$$

It is easy to see (again by transfinite induction on $o(a)$) that $\mathrm{fv}(a)$ indeed is a finite set of variables containing all those free in t_a.

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, 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 $\mathrm{PCF}_{\prec}$ (cf. the beginning of Section 4 below) 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.

A *rank* function for a notation system $\mathscr{T}$ is a function $\mathrm{rk}: T \to \mathbb{N}$ such that (i) for all $a \in T$ and all $n < \mathrm{arity}(\mathrm{rl}(a))$ we have $\mathrm{rk}(a[n]) \leqslant \mathrm{rk}(a)$, and (ii) $\mathrm{lev}(\mathrm{tp}(a[0])) \leqslant \mathrm{rk}(a)$ if $\mathrm{rl}(a) = \mathrm{CAP}$.

3. The extension $\mathscr{T}^*$ of a notation system $\mathscr{T}$

Let $\mathscr{T} = (T, \mathrm{tp}, \mathrm{fv}, \mathrm{rl}, \cdot[\cdot])$ be a notation system. We extend $\mathscr{T}$ by adding for every variable y a constant a_y, for every $n \geqslant 1$ 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 $x_1, \dots, x_n$), 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 constants and function symbols. Using some obvious coding machinery we may clearly assume that T^* is a primitive recursive subset of $\mathbb{N}$, and that $a_i < F(\vec{a})$ for each of the newly introduced function symbols. We now extend the functions tp, fv, rl and $\cdot[\cdot]$ from T to T^*.

3.1. Let us first define $\mathrm{tp}(a)$ for every $a \in T^* \backslash T^*$:

$$\mathrm{tp}(a_y) := \mathrm{Tp}(y),$$

$$\mathrm{tp}(\mathrm{sub}_{\vec{x}}(a, \vec{b})) := \mathrm{tp}(a),$$

$$\mathrm{tp}(\beta(a,b)) := \begin{cases} \rho & \text{if } \mathrm{tp}(a) = [\mathrm{tp}(b) \to \rho], \\ \imath & \text{otherwise,} \end{cases}$$

$$\mathrm{tp}(\mathrm{red}(a)) := \mathrm{tp}(a).$$

We now define the function fv on $T^* \backslash T$:

$$\mathrm{fv}(a_y) := \{y\},$$

$$\mathrm{fv}(\mathrm{sub}_{\vec{x}}(a, \vec{b})) := \bigcup_{z \in \mathrm{fv}(a)} \begin{cases} \mathrm{fv}(b_i) & \text{if } z = x_i, \\ \{z\} & \text{otherwise,} \end{cases}$$

$$\mathrm{fv}(\beta(a,b)) := \mathrm{fv}(a) \cup \mathrm{fv}(b),$$

$$\mathrm{fv}(\mathrm{red}(a)) := \mathrm{fv}(a).$$

Next, we define $\mathrm{rl}(a) \in \textsc{Rules}$ for every $a \in T^* \setminus T$:

$$\mathrm{rl}(a_y) := \textsc{Var}_y,$$

$$\mathrm{rl}(\mathrm{sub}_{\vec{x}}(a,\vec{b})) := \begin{cases} \mathrm{rl}(b_i) & \text{if } \mathrm{rl}(a) = \textsc{Var}_{x_i},\ \mathrm{Tp}(x_i) = \mathrm{tp}(b_i), \\ \textsc{CAp} & \text{if } \mathrm{rl}(a) = \textsc{Ap}_{x_i},\ \mathrm{Tp}(x_i) = \mathrm{tp}(b_i) \\ & \quad \text{and } \mathrm{rl}(b_i) \in \{\textsc{Abs}_y, \textsc{Rep}, \textsc{CAp}\}, \\ \textsc{SeqAp} & \text{if } \mathrm{rl}(a) = \textsc{Ap}_{x_i},\ \mathrm{Tp}(x_i) = \mathrm{tp}(b_i),\ \mathrm{rl}(b_i) = \textsc{Seq}, \\ \textsc{Ap}_y & \text{if } \mathrm{rl}(a) = \textsc{Ap}_{x_i},\ \mathrm{Tp}(x_i) = \mathrm{tp}(b_i),\ \mathrm{rl}(b_i) = \textsc{Var}_y, \\ \mathrm{rl}(b_i) & \text{otherwise, but } \mathrm{rl}(a) = \textsc{Ap}_{x_i} \text{ and } \mathrm{Tp}(x_i) = \mathrm{tp}(b_i), \\ \textsc{Abs}_y & \text{if } \mathrm{rl}(a) = \textsc{Abs}_x, \text{ where } y = v_j^{\mathrm{Tp}(x)}, j \text{ minimal} \\ & \quad \text{such that } y \notin \mathrm{fv}(a) \cup \mathrm{fv}(\vec{b}), \\ \mathrm{rl}(a) & \text{otherwise,} \end{cases}$$

$$\mathrm{rl}(\beta(a,b)) := \begin{cases} \textsc{Rep} & \text{if } \mathrm{rl}(a) \in \{\textsc{Abs}_x, \textsc{Rep}\} \text{ and } \mathrm{tp}(a) = [\mathrm{tp}(b) \to \rho], \\ \textsc{Ap}_x & \text{if } \mathrm{rl}(a) \in \{\textsc{Ap}_x, \textsc{Var}_x\} \text{ and } \mathrm{tp}(a) = [\mathrm{tp}(b) \to \rho], \\ \textsc{CAp} & \text{if } \mathrm{rl}(a) = \textsc{CAp} \text{ and } \mathrm{tp}(a) = [\mathrm{tp}(b) \to \rho], \\ \textsc{SeqAp} & \text{if } \mathrm{rl}(a) = \textsc{Seq} \text{ and } \mathrm{tp}(a) = [\mathrm{tp}(b) \to \rho], \\ \textsc{Const}_0 & \text{otherwise,} \end{cases}$$

$$\mathrm{rl}(\mathrm{red}(a)) := \begin{cases} \textsc{Rep} & \text{if } \mathrm{rl}(a) = \textsc{CAp}, \\ \mathrm{rl}(a) & \text{otherwise.} \end{cases}$$

Clearly, tp, fv and rl are primitive recursive.

Finally we define $a[n] \in T^* \cup \{0\}$ for every $a \in T^* \setminus T$ and $n < \mathrm{arity}(\mathrm{rl}(a))$:

$$a_y[n] := 0,$$

$$\mathrm{sub}_{\vec{x}}(a,\vec{b})[n] := \begin{cases} b_i[n] & \text{if } \mathrm{rl}(a) = \textsc{Var}_{x_i} \text{ and } \mathrm{Tp}(x_i) = \mathrm{tp}(b_i), \\ \mathrm{sub}_{x,\vec{x}'}(a[0], a_y, \vec{b}') & \text{if } \mathrm{rl}(a) = \textsc{Abs}_x, y = v_j^{\mathrm{Tp}(x)}, j \text{ minimal} \\ & \quad \text{s.t. } y \notin \mathrm{fv}(a,\vec{b}), \vec{x}' \text{ is } \vec{x} \text{ without } x, \text{ and} \\ & \quad \vec{b}' \text{ the corresponding subtuple of } \vec{b}, \\ \mathrm{sub}_{\vec{x}}(a[n],\vec{b}) & \text{otherwise,} \end{cases}$$

$$\beta(a,b)[n] := \begin{cases} \mathrm{sub}_x(a[0],b) & \text{if } \mathrm{rl}(a) = \textsc{Abs}_x \text{ and } \mathrm{tp}(a) = [\mathrm{tp}(b) \to \rho], \\ \beta(a[0],b) & \text{if } \mathrm{rl}(a) = \textsc{Rep} \text{ and } \mathrm{tp}(a) = [\mathrm{tp}(b) \to \rho], \\ a & \text{otherwise, but } \mathrm{tp}(a) = [\mathrm{tp}(b) \to \rho] \text{ and } n = 0, \\ b & \text{otherwise, but } \mathrm{tp}(a) = [\mathrm{tp}(b) \to \rho] \text{ and } n = 1, \\ 0 & \text{otherwise,} \end{cases}$$

$$\mathrm{red}(a)[n] := \begin{cases} \beta(\mathrm{red}(a[0]), \mathrm{red}(a[1])) & \text{if } \mathrm{rl}(a) = \textsc{CAp}, \\ \mathrm{red}(a[n]) & \text{otherwise.} \end{cases}$$

It is again easy to see that this is a primitive recursion; recall that sub, β and red on the right-hand side are just function symbols used to generate elements of T^*.

3.2. Let $\mathscr{T}^*$ be the notation system consisting of T^* with the so extended functions tp, fv, rl and $\cdot[\cdot]$. Then we have

Theorem. *If $\mathscr{T}$ is a correct notation system, then so is $\mathscr{T}^*$.*

Proof. We need to show that for all $a \in T^*$ and all $k \in \mathbb{N}$ the conditions (i)–(vi) in the definition of a correct notation system hold. This is done by induction on a. We restrict ourselves to the case red(a).

Case red(a): Recall that

$$\mathrm{tp}(\mathrm{red}(a)) = \mathrm{tp}(a),$$

$$\mathrm{fv}(\mathrm{red}(a)) = \mathrm{fv}(a).$$

Subcase $\mathrm{rl}(a) = \textsc{CAp}$: By definition we then have

$$\mathrm{rl}(\mathrm{red}(a)) = \textsc{Rep},$$

$$\mathrm{red}(a)[0] = \beta(\mathrm{red}(a[0]), \mathrm{red}(a[1])).$$

We must check (vi), i.e.

$$\mathrm{tp}(a) = \mathrm{tp}(\beta(\mathrm{red}(a[0]), \mathrm{red}(a[1]))),$$

$$\mathrm{fv}(\beta(\mathrm{red}(a[0]), \mathrm{red}(a[1]))) \subseteq \mathrm{fv}(a).$$

Since $\mathrm{rl}(a) = \textsc{CAp}$, by IH for a we have $\mathrm{tp}(a[0]) = [\mathrm{tp}(a[1]) \to \mathrm{tp}(a)]$ and $\mathrm{fv}(a[0]) \cup \mathrm{fv}(a[1]) \subseteq \mathrm{fv}(a)$. So both claims follow.

Subcase otherwise. By definition then

$$\mathrm{rl}(\mathrm{red}(a)) = \mathrm{rl}(a),$$

$$\mathrm{red}(a)[n] = \mathrm{red}(a[n]).$$

Therefore, the claim follows immediately from the IH for a (one needs to distinguish cases on $\mathrm{rl}(a)$ here). $\square$

The following theorem states that every α-norm on $\mathcal{T}$ can be extended to an $\varepsilon(\alpha)$-norm on $\mathcal{T}^*$; $\varepsilon(\alpha)$ is the least epsilon number bigger than α. An important point here is that we will not need transfinite induction to prove this theorem. Rather, we will be able to prove it by ordinary induction on a.

Note also that we need to define $o(\mathrm{red}(a)) = 4^{o(a)}$ rather than $3^{o(a)}$ as one might expect from the corresponding estimate for $|\mathrm{R{\scriptsize ED}}(t)|$. The reason will be apparent in case $\mathrm{red}(a)$ below.

3.3. Theorem. *Every norm on $\mathcal{T}$ can be extended to a norm on $\mathcal{T}^*$ by defining*

$$o(a_y) := 0,$$

$$o(\mathrm{sub}_{\vec{x}}(a, \vec{b})) := (\max o(\vec{b})) + o(a),$$

$$o(\beta(a, b)) := o(b) + o(a) + 1,$$

$$o(\mathrm{red}(a)) := 4^{o(a)}.$$

Proof. By induction on a we prove $o(a[n]) < o(a)$ for all $n < \mathrm{arity}(\mathrm{rl}(a))$. Again we restrict ourselves to the case $\mathrm{red}(a)$.

Case $\mathrm{red}(a)$: If $\mathrm{rl}(a) \neq \mathrm{C{\scriptsize AP}}$, then $o(\mathrm{red}(a)[n]) = o(\mathrm{red}(a[n])) = 4^{o(a[n])} < 4^{o(a)} = o(\mathrm{red}(a))$, using the IH for a. So we can assume $\mathrm{rl}(a) = \mathrm{C{\scriptsize AP}}$. Recall that then we have

$$\mathrm{rl}(\mathrm{red}(a)) = \mathrm{R{\scriptsize EP}},$$

$$\mathrm{red}(a)[0] = \beta(\mathrm{red}(a[0]), \mathrm{red}(a[1])).$$

So we obtain

$$\begin{aligned} o(\mathrm{red}(a)[0]) &= o(\beta(\mathrm{red}(a[0]), \mathrm{red}(a[1]))) \\ &= o(\mathrm{red}(a[1])) + o(\mathrm{red}(a[0])) + 1 \\ &= 4^{o(a[1])} + 4^{o(a[0])} + 1 \\ &< 4^{\max(o(a[0]), o(a[1]))+1} \quad \text{(here 4 is needed, for } <) \\ &\leqslant 4^{o(a)} \quad \text{by IH for } a \\ &= o(\mathrm{red}(a)). \quad \square \end{aligned}$$

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

$$\mathrm{rk}(a_y) := 0,$$

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

Proof. We must show that for all $a \in T^*$ and all $n < \mathrm{arity}(\mathrm{rl}(a))$

$$\mathrm{rk}(a[n]) \leqslant \mathrm{rk}(a) \tag{1}$$

and in case $\mathrm{rl}(a) = \textsc{CAp}$ also

$$\mathrm{lev}(\mathrm{tp}(a[0])) \leqslant \mathrm{rk}(a). \tag{2}$$

This is done by induction on a. Again we restrict ourselves to the case $\mathrm{red}(a)$.

Case $\mathrm{red}(a)$: *Subcase* $\mathrm{rl}(a) = \textsc{CAp}$. Then by definition $\mathrm{rl}(\mathrm{red}(a)) = \textsc{Rep}$ and $\mathrm{red}(a)[0] = \beta(\mathrm{red}(a[0]), \mathrm{red}(a[1]))$. Hence,

$$\begin{aligned}\mathrm{rk}(\mathrm{red}(a)[0]) &= \mathrm{rk}(\beta(\mathrm{red}(a[0]), \mathrm{red}(a[1])))\\ &= \max(\mathrm{lev}(\mathrm{tp}(a[1])), \mathrm{rk}(\mathrm{red}(a[0])), \mathrm{rk}(\mathrm{red}(a[1])))\\ &= \max(\mathrm{lev}(\mathrm{tp}(a[1])), \mathrm{rk}(a[0]) \dot{-} 1, \mathrm{rk}(a[1]) \dot{-} 1)\\ &\leqslant \mathrm{rk}(a) \dot{-} 1,\end{aligned}$$

since by IH for a we have $\mathrm{rk}(a[n]) \leqslant \mathrm{rk}(a)$ and $\mathrm{lev}(\mathrm{tp}(a[0])) \leqslant \mathrm{rk}(a)$, hence $\mathrm{lev}(\mathrm{tp}(a[1])) \leqslant \mathrm{rk}(a) \dot{-} 1$. The second claim (2) does not apply, since $\mathrm{rl}(\mathrm{red}(a)) = \textsc{Rep} \neq \textsc{CAp}$.

Subcase otherwise. By definition $\mathrm{rl}(\mathrm{red}(a)) = \mathrm{rl}(a)$ and $\mathrm{red}(a)[n] = \mathrm{red}(a[n])$. Hence,

$$\begin{aligned}\mathrm{rk}(\mathrm{red}(a)[n]) &= \mathrm{rk}(\mathrm{red}(a[n]))\\ &= \mathrm{rk}(a[n]) \dot{-} 1\\ &\leqslant \mathrm{rk}(a) \dot{-} 1 \quad \text{by IH for } a\\ &= \mathrm{rk}(\mathrm{red}(a)).\end{aligned}$$

The second claim (2) again does not apply, since $\mathrm{rl}(\mathrm{red}(a))$ can never be $\textsc{CAp}$. □

4. First application: the trade-off theorem

Here we work with $\textsc{Pcf}_\alpha$ (cf. [10, 9]), a partial version of Gödel's T with transfinitely bounded higher-type fixed point operators $\mathrm{Y}_{\rho,\alpha}$ instead of primitive recursion operators. The details of $\textsc{Pcf}_\alpha$ are not important for our present purpose to demonstrate how one can work with infinite expansions of finite terms. However, for the convenience of the reader we repeat the definition of the bounded higher-type fixed point operators. $\mathrm{Y}_{\rho,\alpha}$ is to be an object of type $[(\iota \to \rho) \to \iota \to \rho] \to \iota \to \rho$. We define $\mathrm{Y}_{\rho,\alpha}(G,x,\vec{x})$ by (transfinite) α-recursion on x, as follows. If x is undefined, then $\mathrm{Y}_{\rho,\alpha}(G,x,\vec{x})$ is undefined. Otherwise, assume that for all y with $y \prec x$ and all $\vec{y}$ the value $\mathrm{Y}_{\rho,\alpha}(G,y,\vec{y})$ is already known. Then let

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

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

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

For simplicity we do not use measure functions in PCF_α as in [10], and allow $\mathrm{Y}_{\rho,\alpha}$ only in a context $\mathrm{Y}_{\rho,\alpha}N$ with N closed (see [9] for a proof that this is not a restriction).

4.1. In order to obtain a notation system for PCF_α-terms we need to introduce additional constructs which serve to denote approximations of functionals defined by recursion. We want to define for any such term a notation for its infinite expansion. Clearly, there is no problem in all cases except $\mathrm{Y}N$; there we will apply induction on the number of nestings of Y.

Let us define now the notation system $\mathscr{P} = (P, \mathrm{tp}, \mathrm{fv}, \mathrm{rl}, \cdot[\cdot])$. We assume that some constants denoting primitive recursive functions are given, where each such has an arity $n \geqslant 0$; numerals $\underline{k}$ for each $k \in \mathbb{N}^\perp$ should always be present. $\mathscr{P}$ is defined recursively by the following clauses:

(i) For every variable x of type ρ we have $x \in P$, $\mathrm{tp}(x) = \rho$, $\mathrm{fv}(x) = \{x\}$ and $\mathrm{rl}(x) = \mathrm{VAR}_x$.

(ii) For every function constant c of arity n and $N_1, \dots, N_1 \in P$ we have $cN_1 \dots N_n \in P$ and

$$\mathrm{tp}(cN_1 \cdots N_n) := \iota,$$

$$\mathrm{fv}(cN_1 \cdots N_n) := \mathrm{fv}(N_1) \cup \cdots \cup \mathrm{fv}(N_n),$$

$$\mathrm{rl}(cN_1 \cdots N_n) := \mathrm{CONST}_c,$$

$$(cN_1 \cdots N_n)[i] := N_{i+1} \quad \text{for } i < n.$$

In particular, for every $k \in \mathbb{N}^\perp$ we have $\underline{k} \in P$, $\mathrm{tp}(\underline{k}) = \iota$, $\mathrm{fv}(\underline{k}) = \emptyset$ and $\mathrm{rl}(\underline{k}) = \mathrm{CONST}_k$.

(iii) For every $M \in P$ with type $\mathrm{tp}(M) = \sigma$ and every variable x of type ρ we have $\lambda x M \in P$ and

$$\mathrm{tp}(\lambda x M) := \rho \to \sigma,$$
$$\mathrm{fv}(\lambda x M) := \mathrm{fv}(M) \setminus \{x\},$$
$$\mathrm{rl}(\lambda x M) := \textsc{Abs}_x,$$
$$(\lambda x M)[0] := M.$$

(iv) For every $M, N \in P$ with types $\mathrm{tp}(M) = \rho \to \sigma$ and $\mathrm{tp}(N) = \rho$ we have $MN \in P$ and

$$\mathrm{tp}(MN) := \sigma,$$
$$\mathrm{fv}(MN) := \mathrm{fv}(M) \cup \mathrm{fv}(N),$$
$$\mathrm{rl}(MN) := \begin{cases} \textsc{Ap}_x & \text{if } \mathrm{rl}(M) \in \{\textsc{Var}_x, \textsc{Ap}_x\}, \\ \textsc{SeqAp} & \text{if } \mathrm{rl}(M) = \textsc{Seq}, \\ \textsc{CAp} & \text{if } \mathrm{rl}(M) \in \{\textsc{Abs}_x, \textsc{CAp}\}, \end{cases}$$
$$(MN)[0] := M,$$
$$(MN)[1] := N.$$

(v) Let a closed term N of type $(\iota \to \rho) \to \iota \to \rho$ be given. We want to define a notation for $\mathrm{Y}_{\rho,\alpha} N$ and its finite approximations, using induction on the number of nestings of $\mathrm{Y}_{\sigma,\alpha}$. Recall the defining equation

$$\mathrm{Y}_{\rho,\alpha} N \zeta = N [\mathrm{Y}_{\rho,\alpha} N]_{<\zeta} \zeta.$$

Now, consider the term Nux with a new variable u of type $\iota \to \rho$. If we replace in Nux all outermost subterms of the form $\mathrm{Y}_{\sigma_1,\alpha} N_1, \dots, \mathrm{Y}_{\sigma_p,\alpha} N_p$ by variables $z_1, \dots, z_p$ and normalize the resulting term, we obtain a term A which is a finite normal λ-term, built from $u, x, z_1, \dots, z_p$ by λ-abstraction and application, such that we may rewrite this equation in the form

$$\mathrm{Y}_{\rho,\alpha} N \zeta = A_{u,x,\vec{z}}[[\mathrm{Y}_{\rho,\alpha} N]_{<\zeta}, \underline{\zeta}, \mathrm{Y}_{\sigma_1,\alpha} N_1, \dots, \mathrm{Y}_{\sigma_p,\alpha} N_p].$$

By IH we can assume that we already have notations $G_1, \dots, G_p$ for $\mathrm{Y}_{\sigma_1,\alpha} N_1, \dots, \mathrm{Y}_{\sigma_p,\alpha} N_p$. We introduce a new notation F for $\mathrm{Y}_{\rho,\alpha} N$, and moreover, for every subterm B of A infinitely many notations B^A_ζ, intended to denote the value of B with u substituted by $[\mathrm{Y}_{\rho,\alpha} N]_{<\zeta}$, x by $\underline{\zeta}$ and $\vec{z}$ by $\mathrm{Y}_{\sigma_1,\alpha} N_1, \dots, \mathrm{Y}_{\sigma_p,\alpha} N_p$. So A^A_ζ denotes the value of $\mathrm{Y}_{\rho,\alpha} N \underline{\zeta}$. (The upper index A will be needed for the definition of a norm below.) Let us assume for simplicity that the type ρ of $\mathrm{Y}_{\rho,\alpha} N \zeta$ is a ground type; the necessary modifications in the general case are rather obvious. By the explanations above it should be clear how to define tp, fv, rl and $\cdot[\cdot]$

for these constants:

$\mathrm{tp}(B^A_\zeta) := \mathrm{tp}(B)$ (we assume here that the finite term B denotes itself),

$\mathrm{fv}(B^A_\zeta) := \mathrm{fv}(B) \backslash \{u, x, \vec{z}\}$.

$\mathrm{rl}(B^A_\zeta)$ is defined by cases on B. Recall that $u, x, \vec{z}$ we substitute notations for a sequence (the cut-off sequence corresponding to $[\mathrm{Y}_{\rho,\alpha} N]_{<\zeta}$), the constant $\underline{\zeta}$ and the sequence corresponding to $\vec{G}$, respectively. Hence,

$\mathrm{rl}(u^A_\zeta) := \textsc{Seq}$,

$\mathrm{rl}((uC)^A_\zeta) := \textsc{SeqAp}$,

$\mathrm{rl}(x^A_\zeta) := \textsc{Const}_\zeta$,

$\mathrm{rl}((z_k)^A_\zeta) := \textsc{Seq}$,

$\mathrm{rl}((z_k C)^A_\zeta) := \textsc{SeqAp}$,

$\mathrm{rl}(B^A_\zeta) := \mathrm{rl}(B)$ otherwise.

Also, for $B^A_\zeta[i]$ we need to distinguish cases on B. So for $i < \mathrm{arity}(\mathrm{rl}(B^A_\zeta))$ we define $B^A_\zeta[i]$ by

$$u^A_\zeta[i] := \begin{cases} A^A_\eta & \text{if } i = \eta < \zeta, \\ \textsc{Const}_\perp & \text{otherwise,} \end{cases}$$

$(z_k)^A_\zeta[i] := G_k[i]$,

$(z_k C)^A_\zeta[0] := G_k$,

$B^A_\zeta[i] := (B[i])^A_\zeta$ otherwise.

Finally, for the notation F of $\mathrm{Y}_{\rho,\alpha} N$ we define

$\mathrm{tp}(F) := \iota \to \rho$,

$\mathrm{fv}(F) := \emptyset$,

$\mathrm{rl}(F) := \textsc{Seq}$,

$F[\zeta] := A^A_\zeta$.

In the general case of a composed ρ we need to modify this in the obvious way according to the view of $\langle s_i \rangle_{i \in \mathbb{N}}$ as an abbreviation of $\lambda x \vec{x} \langle s_i \vec{x} \rangle_{i \in \mathbb{N}} x$.

Via some obvious coding we may assume that P is a subset of $\mathbb{N}$. Note that the repetition rule is not used in $\mathscr{P}$ (i.e. $\mathrm{rl}(M) \neq \textsc{Rep}$ for all $M \in P$).

Proposition. *The notation system $\mathscr{P}$ is correct and there exist a primitive recursive rank function and a primitive recursive $\alpha \cdot \omega$-norm o on P.*

Proof. Correctness is easy, by checking clauses (i)–(v) in the definition of a correct notation system. Also, a rank function for $\mathscr{P}$ can be defined easily:

$$\begin{aligned}
&\mathrm{rk}(x) := 0,\\
&\mathrm{rk}(cN_1\cdots N_n) := \max(\mathrm{rk}(N_1),\dots,\mathrm{rk}(N_n)),\\
&\mathrm{rk}(\lambda xM) := \mathrm{rk}(M),\\
&\mathrm{rk}(M^\rho N) := \begin{cases} \max(\mathrm{lev}(\rho),\mathrm{rk}(M),\mathrm{rk}(N)) & \text{if } \mathrm{rl}(M)=\mathrm{CAP},\\ \max(\mathrm{rk}(M),\mathrm{rk}(N)) & \text{otherwise}\end{cases}
\end{aligned}$$

and

$$\mathrm{rk}(B_\zeta^A) := \mathrm{rk}(F) := \max(\mathrm{lev}(\rho_1),\dots,\mathrm{lev}(\rho_p),\mathrm{lev}(\rho)),$$

where $\rho,\rho_1,\dots,\rho_p$ are the types F and the auxiliary functionals $G_1,\dots,G_p$ used in its definition.

Concerning the $\alpha\cdot\omega$-norm o we restrict attention to α-recursion. Let F be defined by α-recursion from $G_1,\dots,G_p$ and assume $o(G_k)\leqslant\alpha\ell$. Then define

$$o(B_\zeta^A) := \alpha\ell + (|A|+1)\zeta + |B| + 1,$$

where $|A|$ denotes the complexity of the finite term A (cf. Section 1). We check that this indeed defines a norm. For $i<\mathrm{arity}(\mathrm{rl}(B_\zeta^A))$ we have

$$\begin{aligned}
o(x_\zeta^A[i]) &= o(A_\eta^A) \quad \text{if } i=\eta<\zeta\\
&= \alpha\ell + (|A|+1)\eta + |A| + 1\\
&\leqslant \alpha\ell + (|A|+1)\zeta \quad \text{since } \eta<\zeta\\
&< o(x_\zeta^A),
\end{aligned}$$

$$o((z_k)_\zeta^A[i]) = o(G_k[i]) <_{\mathrm{IH}} o(G_k) \leqslant \alpha\ell < o((z_k)_\zeta^A),$$

$$o((z_kC)_\zeta^A[0]) = o(G_k) \leqslant \alpha\ell < o((z_kC)_\zeta^A)$$

and for all other cases

$$\begin{aligned}
o(B_\zeta^A[i]) &= o(B[i]_\zeta^A)\\
&= \alpha\ell + (|A|+1)\zeta + |B[i]| + 1\\
&< \alpha\ell + (|A|+1)\zeta + |B| + 1\\
&= o(B_\zeta^A).
\end{aligned}$$

Finally, we have to define $o(F)$ such that $o(F[\zeta]) = o(A_\zeta^A) < o(F)$ for all $\zeta<\alpha$. But this is true if we define

$$o(F) := \alpha\ell + (|A|+1)\alpha + 1. \qquad \square$$

As described in Section 3 we can now extend $\mathscr{P}$ to a correct notation system $\mathscr{P}^* = (P^*, \ldots)$ with primitive recursive rank function $\mathrm{rk}: P^* \to \mathbb{N}$ and norm $o: P^* \to \varepsilon(\alpha)$.

4.2. *Coding finite sequences.* Note that we do not assume the existence of pairing/unpairing functions π, π_1, π_2 at ground types. The reason is simply that in the model of partial continuous functionals such functions do not exist. To see this, assume we have a continuous pairing π. Then $\pi(\bot, k) =: n_k \neq \bot$ for some $k \in \mathbb{N}$, since π is injective. But by monotonicity we then have $\pi(m,k) = n_k$ for all m, contradicting the injectivity of π.

However, we can trivially encode a finite sequence $u_0, \ldots, u_{m-1}$ of objects of the same type τ into a function w of type $\iota \to \tau$, by defining $w(i)$ to be u_i for $i < m$ and arbitrarily otherwise. Note that the level of $\iota \to \tau$ is the same as the level of τ, provided the latter is $\geqslant 1$.

4.3. *Transformation functionals.* Let objects $u_1, \ldots, u_m$ of types $\tau_1, \ldots, \tau_m$ be given. Later – when defining codes for assignments – we will need to transform these objects into others of a common type τ, in such a way that the original objects u_i can be recovered. To this end we define *transformation functionals* and their inverses. Their construction is relative to the finite set $S = \{\tau_1, \ldots, \tau_m\}$ of types. For $\tau_i =: \vec{\tau}_i \to \iota$ we define

$$\tau := \tau_S := \vec{\tau}_1 \to \vec{\tau}_2 \to \cdots \vec{\tau}_m \to \iota$$

and transformation functionals $\mathrm{Tr}^{\tau}_{\tau_i}$ of type $\tau_i \to \tau$, $\mathrm{Tr}^{\tau_i}_{\tau}$ of type $\tau \to \tau_i$ by

$$\mathrm{Tr}^{\tau}_{\tau_i}(u)(\vec{v}_1, \ldots, \vec{v}_m) := u(\vec{v}_i),$$

$$\mathrm{Tr}^{\tau_i}_{\tau}(w)(\vec{v}_i) := w(\vec{0}, \ldots, \vec{0}, \vec{v}_i, \vec{0}, \ldots, \vec{0}).$$

With this definition we clearly have $\mathrm{Tr}^{\tau_i}_{\tau}(\mathrm{Tr}^{\tau}_{\tau_i}(u)) = u$, since

$$\mathrm{Tr}^{\tau_i}_{\tau}(\mathrm{Tr}^{\tau}_{\tau_i}(u))(\vec{v}_i) = \mathrm{Tr}^{\tau}_{\tau_i}(\vec{0}, \ldots, \vec{0}, \vec{v}_i, \vec{0}, \ldots, \vec{0}) = u(\vec{v}_i).$$

Similar transformation functionals have been introduced by Gandy [2]. However, for his construction it is necessary to have pairing/unpairing functions at ground types, which we do not assume here (cf. Section 4.2).

4.4. *Valuation functionals.* Consider a notation system $\mathscr{P}$ with $\prec$-norm o. Let $a \in P$ be a notation of a closed term t with type ρ. Then we want to define the value $\mathrm{VAL}_\rho(a)$ of t. However, for the definition of the VAL_ρ 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 VAL_ρ, 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, of types $\tau_1, \dots, \tau_m$, respectively, in each case. 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, as described in Section 4.3, transform all the u_i to objects of 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. $\mathrm{Tr}^{\tau_i}_{\tau}(\mathrm{Tr}^{\tau}_{\tau_i}(u)) = u$). With these $\mathrm{Tr}^{\tau}_{\tau_i}(u_i)$ build as in Section 4.2 a function w of type $\iota \to \tau$ such that $w(a_i) = \mathrm{Tr}^{\tau}_{\tau_i}(u_i)$. From w one can then read off the functionals u_i assigned to x_i by $u_i = \mathrm{Tr}^{\tau_i}_{\tau}(w(a_i))$. To an extension (or update) of the given assignment by the requirement that the functional u should be assigned to the variable x of type τ_i and denoted by $a \in P$ there corresponds a change of the code w to $\textsc{Ext}(a, \mathrm{Tr}^{\tau}_{\tau_i}(u), w)$ with a functional $\textsc{Ext}$ that can be defined easily by

$$\textsc{Ext}(a,u,w)(b,\vec{v}_1,\dots,\vec{v}_m) := \begin{cases} \mathrm{Tr}^{\tau_i}_{\tau}(u)(\vec{v}_i) & \text{if } a = b \text{ and } \mathrm{tp}(a) = \tau_i, \\ w(b,\vec{v}_1,\dots,\vec{v}_m) & \text{otherwise.} \end{cases}$$

We now define the valuation functionals $\textsc{Val}_\rho$ ($\rho \in S$) with respect to a fixed finite set S of types and our given notation system $\mathscr{P}$ with $\prec$-norm o. Let w be a variable of type $\tau := \tau_S$. Then

$$\textsc{Val}_\rho(a,w)$$

$$:= \begin{cases} \mathrm{Tr}^{\rho}_{\tau}(w(a)) & \text{if } \mathrm{rl}(a) = \textsc{Var}_x \text{ and } \mathrm{tp}(a) = \rho, \\ f(\textsc{Val}_\iota(a[0],w),\dots, & \\ \quad \textsc{Val}_\iota(a[n-1],w)) & \text{if } \mathrm{rl}(a) = \textsc{Const}_c \text{ and } c \text{ codes } f, \\ \lambda v^{\sigma}\textsc{Val}_{\tau}(a[0],\textsc{Ext}(a_x,\mathrm{Tr}^{\tau}_{\sigma}(v),w)) & \text{if } \mathrm{rl}(a) = \textsc{Abs}_x \text{ and} \\ & \quad \mathrm{tp}(a) = \sigma \to \sigma' = \rho, \\ \textsc{Val}_{\sigma\to\rho}(a[0],w)(\textsc{Val}_\sigma(a[1],w)) & \text{if } \mathrm{rl}(a) \in \{\textsc{Ap}_x, \textsc{SeqAp}, \textsc{CAp}\}, \\ & \quad \mathrm{tp}(a) = \rho, \quad \mathrm{tp}(a[0]) = \sigma \to \rho \\ & \quad \text{and } \mathrm{tp}(a[1]) = \sigma, \\ \lambda u^{\iota}\textsc{Val}_\iota(a[u],w) & \text{if } \mathrm{rl}(a) = \textsc{Seq} \text{ and } \mathrm{tp}(a) = \iota \to \iota = \rho, \\ \textsc{Val}_\rho(a[0],w) & \text{if } \mathrm{rl}(a) = \textsc{Rep} \text{ and } \mathrm{tp}(a) = \rho, \\ 0^{\rho} & \text{otherwise.} \end{cases}$$

It might be helpful to formulate the essential properties of the $\textsc{Val}_\rho$-functionals in a slightly different notation. Let $\ulcorner \vec{x} \mapsto \vec{u} \urcorner$ denote the code of the assignment $(x_1 \mapsto u_1, \dots, x_m \mapsto u_m)$ constructed as described above; then $\ulcorner \vec{x}, x \mapsto \vec{u}, u \urcorner$ denotes $\textsc{Ext}(a_x, \mathrm{Tr}^{\tau}_{\sigma}(u), \ulcorner \vec{x} \mapsto \vec{u} \urcorner)$. We have

$$\textsc{Val}_{\rho_i}(a_{x_i}, \ulcorner \vec{x} \mapsto \vec{u} \urcorner) = u_i,$$

$$\textsc{Val}_\iota(a, \ulcorner \vec{x} \mapsto \vec{u} \urcorner) = f(\textsc{Val}_\iota(a[0], \ulcorner \vec{x} \mapsto \vec{u} \urcorner), \dots, \textsc{Val}_\iota(a[n-1], \ulcorner \vec{x} \mapsto \vec{u} \urcorner))$$

$$\text{if } \mathrm{rl}(a) = \textsc{Const}_c \text{ and } c \text{ codes } f,$$

$$\mathrm{VAL}_{\sigma\to\sigma'}(a, \ulcorner \vec{x} \mapsto \vec{u} \urcorner)(u) = \mathrm{VAL}_{\sigma'}(a[0], \ulcorner \vec{x}, x \mapsto \vec{u}, u \urcorner)$$
$$\text{if } \mathrm{rl}(a) = \mathrm{ABS}_x \text{ and } \mathrm{tp}(a) = \sigma \to \sigma' = \rho,$$

$$\mathrm{VAL}_{\rho}(a, \ulcorner \vec{x} \mapsto \vec{u} \urcorner) = \mathrm{VAL}_{\sigma\to\rho}(a[0], \ulcorner \vec{x} \mapsto \vec{u} \urcorner)(\mathrm{VAL}_{\sigma}(a[1], \ulcorner \vec{x} \mapsto \vec{u} \urcorner))$$
$$\text{if } \mathrm{rl}(a) \in \{\mathrm{AP}_x, \mathrm{SEQAP}, \mathrm{CAP}\},$$
$$\mathrm{tp}(a) = \rho,\ \mathrm{tp}(a[0]) = \sigma \to \rho \text{ and } \mathrm{tp}(a[1]) = \sigma,$$

$$\mathrm{VAL}_{\iota\to\iota}(a, \ulcorner \vec{x} \mapsto \vec{u} \urcorner)(u) = \mathrm{VAL}_{\iota}(a[u], \ulcorner \vec{x} \mapsto \vec{u} \urcorner)$$
$$\text{if } \mathrm{rl}(a) = \mathrm{SEQ} \text{ and } \mathrm{tp}(a) = \iota \to \iota = \rho,$$

$$\mathrm{VAL}_{\rho}(a, \ulcorner \vec{x} \mapsto \vec{u} \urcorner) = \mathrm{VAL}_{\rho}(a[0], \ulcorner \vec{x} \mapsto \vec{u} \urcorner)$$
$$\text{if } \mathrm{rl}(a) = \mathrm{REP} \text{ and } \mathrm{tp}(a) = \rho.$$

Note that for fixed $\beta \leqslant \alpha$ this can be viewed as a definition in PCF_α: the finitely many functionals $\mathrm{VAL}_\rho = \mathrm{VAL}_\rho^{\beta,S}$ $(\rho \in S)$ are defined by simultaneous β-recursion, where all auxiliary functionals (in particular $\cdot[\cdot]$) are primitive recursive.

If $a \in P$ with norm $o(a) < \beta$ codes the infinite term t_a^ρ all of whose subterms have types from the finite set S, then for every assignment $\vec{x} \mapsto \vec{u}$

$$\mathrm{VAL}_\rho^{\beta,S}(a, \ulcorner \vec{x} \mapsto \vec{u} \urcorner) = [\![t_a]\!](x_1 \mapsto u_1, \ldots, x_m \mapsto u_m);$$

this is proved by β-induction on a.

4.5. 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* $\mathrm{lev}(\rho) \leqslant n + m + 1$ *with* $m \geqslant 1$. *Then one can find* $\beta < 4_m(\alpha\cdot\omega)$ *and a closed term M' of* PCF_β *involving just one* $\mathrm{Y}_{\alpha,\beta}$ *such that* $[\![M]\!] = [\![M']\!]$ *and* $\mathrm{lev}(\sigma) \leqslant n + 1$.

Proof. 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. As described in Section 4.1, construct a correct notation system $\mathscr{P}$ for PCF_α, and also a primitive recursive rank function and an $\alpha\cdot\omega$-norm for $\mathscr{P}$. Now, extend $\mathscr{P}$ to a correct notation system $\mathscr{P}^*$, as described in Section 3, again with a primitive recursive rank function and an $\varepsilon(\alpha)$-norm. In $\mathscr{P}^*$ we can form $\mathrm{red}^m(M)$. Since we have $\mathrm{rk}(M) = \mathrm{lev}(\rho) \leqslant n + m + 1$ in $\mathscr{P}$, from Section 3.4 we know that $\mathrm{rk}(\mathrm{red}^m(M)) \leqslant n + 1$ in $\mathscr{P}^*$, hence all bound variables in the term denoted by $\mathrm{red}^m(M)$ have levels $\leqslant n$. Concerning the norm in $\mathscr{P}$ we have $o(M) < \alpha\cdot k < \alpha\cdot\omega$ for some $k < \omega$, hence in $\mathscr{P}^*$

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

Let S be the finite set of all types used in the term M, and $S_k := \{\rho \in S \mid \mathrm{lev}(\rho) \leqslant k\}$. As described in Section 4.4 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. □

5. Second application: Continuous normalization

It was observed by Mints [4] that the usual cut-elimination operator also works for non well-founded derivations. Moreover – as stressed by Kreisel – this more general setup makes it impossible to define the cut elimination operator by transfinite recursion, and hence one is forced into a closer analysis revealing the fact that one can give a primitive recursive definition of such an operator.

It is well known that for natural deduction systems a primitive recursive (and hence continuous) normalization operator can be defined similarly. In [8] this has been done for a standard set of rules including permutative conversions. Ruckert in his thesis [5] developed a rather general notion of natural deduction systems, which e.g. includes Martin–Löf type theories. For an abstract formulation of conversion rules he could construct a continuous normalization operator. However, in both cases the definition was given in somewhat informal terms; at some points (in particular [8, p. 346; 5, p. 71]), one had to resort to a rather pictorial style to explain what was going on.

Here we show that Buchholz' formal treatment of continuous cut elimination in [1] can be adapted to a natural deduction setting. This allows an explicit construction of the normalization operator, in a form suitable for further (e.g. metamathematical) analysis. In particular, an explicit and simple definition of a modulus of continuity can be given, as for cut elimination in [1]. In the present paper we only carry this out for a non well-founded version of the system of infinite terms of Section 1. It turns out that the presence of free (assumption) variables does not create essential new difficulties once we restrict attention to infinite terms with finite sets of free variables at each node (as in [6]). However, the approach is perfectly general and could, for instance, be applied to Ruckert's setting as well.

In order to treat continuous normalization we make use of special, possibly non-well-founded notation systems. These are based on a function $\varphi : \mathbb{N}^{<\omega} \to \mathbb{N}$, which is thought of as coding an infinite not necessarily well-founded term. The notations are just codes $\sigma = \langle n_0, \dots, n_{k-1} \rangle$ of finite sequences of natural numbers, so the predecessor function can simply be defined by $\sigma[n] := \sigma * \langle n \rangle$. For each node $\sigma = \langle n_0, \dots, n_{k-1} \rangle \in \mathbb{N}^{<\omega}$ the value $\varphi(\sigma)$ codes the type $\text{tp}_\varphi(\sigma) := (\varphi(\sigma))_0$, the finite set of free variables $\text{fv}_\varphi(\sigma) := (\varphi(\sigma))_1$ and the rule $\text{rl}_\varphi(\sigma) := (\varphi(\sigma))_2$ at σ. If such a notation system $\mathcal{T}_\varphi := (\mathbb{N}^{<\omega}, \text{tp}_\varphi, \text{fv}_\varphi, \text{rl}_\varphi, \cdot[\cdot])$ is correct, then it clearly can be thought of as a representation of the possibly non well-founded term given by φ. We now extend $\mathcal{T}_\varphi$ as described in Section 3 to $(\mathcal{T}_\varphi)^* := ((\mathbb{N}^{<\omega})^*, \text{tp}_\varphi, \text{fv}_\varphi, \text{rl}_\varphi, \cdot[\cdot]_\varphi)$. This should be viewed as a system of notations for *many* terms (as opposed to $\mathcal{T}_\varphi$, which only has notations for the subterms of the single possibly non well-founded term given by φ): every $a \in (\mathbb{N}^{<\omega})^*$ denotes a (again possibly non well-founded) term, whose label at node σ is given by $a[\sigma]_\varphi := a[n_0]_\varphi \dots [n_{k-1}]_\varphi$ for $\sigma = \langle n_0, \dots, n_{k-1} \rangle$, or more precisely by the three items

$\mathrm{tp}_\varphi(a[\sigma]_\varphi)$, $\mathrm{fv}_\varphi(a[\sigma]_\varphi)$ and $\mathrm{rl}_\varphi(a[\sigma]_\varphi)$. Since the term described by φ is denoted in $\mathscr{T}_\varphi$ by ε (the root node), $a = \mathrm{red}(\varepsilon)$ denotes in $\mathscr{T}_\varphi^*$ the result of reducing the rank of this term by one. Therefore, we have

5.1. Theorem (Continuous normalization). *Let φ be a (not necessarily well-founded) infinite term of rank $\leqslant k+1$. Then $R(\varphi) := \lambda\sigma(\mathrm{red}(\varepsilon))[\sigma]_\varphi$ is the result of reducing the rank of this term by one. In particular, if φ_t represents $t \in \mathrm{T{\small ERMS}}^\circ$, then $R(\varphi_t)$ represents $\mathrm{R{\small ED}}(t)$. Moreover, if φ and ψ coincide on N_σ, where*

$$N_{\langle n_0,\dots,n_{k-1}\rangle} := \{\langle m_0,\dots,m_{l-1}\rangle \mid l \leqslant k \wedge \{m_0,\dots,m_{l-1}\} \subseteq \{0,1,n_0,\dots,n_{k-1}\}\},$$

then $R(\varphi)(\sigma) = R(\psi)(\sigma)$, i.e. $\lambda\sigma N(\sigma)$ is a modulus of continuity for R.

Proof. The first two propositions are immediate consequences of the general arguments in Section 3 (Theorems 3.2–3.4). To see that $\lambda\sigma N(\sigma)$ is a modulus of continuity for R, define $\mathrm{C}(a)$ as the set of all $\sigma \in \mathbb{N}^{<\omega}$ that ‘occur’ in a:

$$\mathrm{C}(\sigma) := \{\sigma\} \quad \text{for } \sigma \in \mathbb{N}^{<\omega},$$

$$\mathrm{C}(a_y) := \mathrm{C}(0) := \emptyset,$$

$$\mathrm{C}(\mathrm{sub}_{\vec{x}}(a, \vec{b})) := \mathrm{C}(a) \cup \mathrm{C}(\vec{b}),$$

$$\mathrm{C}(\beta(a,b)) := \mathrm{C}(a) \cup \mathrm{C}(b),$$

$$\mathrm{C}(\mathrm{red}(a)) := \mathrm{C}(a).$$

Then for all $a \in (\mathbb{N}^{<\omega})^*$ we have $\mathrm{C}(a[n]_\varphi) \subseteq \mathrm{C}(a) * N_{\langle n\rangle}$; this can be proved easily by induction on a. Hence, $\mathrm{C}(a[\sigma]_\varphi) \subseteq \mathrm{C}(a) * N_\sigma$ for all $\sigma \in \mathbb{N}^{<\omega}$.

Now let φ, $\psi : \mathbb{N}^{<\omega} \to \mathbb{N}$ and $a \in (\mathbb{N}^{<\omega})^*$. Assume $\varphi \restriction \mathrm{C}(a) = \psi \restriction \mathrm{C}(a)$. Then by induction on a one proves easily that $\mathrm{tp}_\varphi(a) = \mathrm{tp}_\psi(a)$ and $\mathrm{fv}_\varphi(a) = \mathrm{fv}_\psi(a)$, and using these two facts also $\mathrm{rl}_\varphi(a) = \mathrm{rl}_\psi(a)$. From all three equalities one finally proves $\forall n(a[n]_\varphi = a[n]_\psi)$, again by induction on a. Hence, for every $\sigma \in \mathbb{N}^{<\omega}$, if $\varphi \restriction (\mathrm{C}(a) * N_\sigma) = \psi \restriction (\mathrm{C}(a) * N_\sigma)$, then $a[\sigma]_\varphi = a[\sigma]_\psi$; this follows easily from the above by induction on σ. □

Acknowledgements

I would like to thank Wilfried Buchholz, Peter Clote, Roy Dyckhoff, Karl-Heinz Niggl, Stan Wainer and an anonymous referee for their helpful comments.

References

[1] W. Buchholz, Notation systems for infinitary derivations, Arch. Math. Logic 30 (1991) 277–296.

[2] R.O. Gandy, Computable functionals of finite type I, in: J. Crossley (Ed.), Sets, Models and Recursion Theory, North-Holland, Amsterdam, 1967, pp. 202–242.

[3] G. Kreisel, On the interpretation of non-finitist proofs II, J. Symbolic Logic 17 (1952) 43–58.
[4] G.E. Mints, Finite investigations of transfinite derivations, J. Sov. Math. 10 (1978) 548–596. Trans. from: Zap. Nauchn. Semin. LOMI 49, 1975.
[5] M. Ruckert, Church-Rosser Theorem und Normalisierung für Termkalküle mit unendlichen Termen unter Einschluß permutativer Reduktionen, Ph.D. Thesis, Mathematisches Institut der Universität München, 1985.
[6] H. Schwichtenberg, Elimination of higher type levels in definitions of primitive recursive functionals by means of transfinite recursion, in: H.E. Rose, J.C. Shepherdson (Eds.), Logic Colloq. '73, North-Holland, Amsterdam, 1975, pp. 279–303.
[7] H. Schwichtenberg, Proof theory: some applications of cut-elimination, in: J. Barwise (Ed.), Handbook of Math. Logic, vol. 90, Studies in Logic and the Foundations of Mathematics, chapter Proof Theory and Constructive Mathematics, North-Holland, Amsterdam, 1977, pp. 867–895.
[8] H. Schwichtenberg, Ein einfaches Verfahren zur Normalisierung unendlicher Herleitungen, in: E. Börger (Ed.), Computation Theory and Logic, Lecture Notes in Computer Science, vol. 270, Springer, Berlin, 1986, pp. 334–348.
[9] H. Schwichtenberg, Classifying recursive functions. in: E. Griffor (Ed.), Handbook of Recursion Theory, North-Holland, Amsterdam, to appear.
[10] H. Schwichtenberg, S.S. Wainer, Ordinal bounds for programs, in: P. Clote, J. Remmel (Eds.), Feasible Mathematics II, Birkhäuser, Boston, 1995, pp. 387–406.
[11] W.W. Tait, Infinitely long terms of transfinite type, in: J. Crossley, M. Dummett (Eds.), Formal Systems and Recursive Functions, North-Holland, Amsterdam, 1965, pp. 176–185.
[12] A. Weiermann, How is it that infinitary methods can be applied to finitary mathematics? Gödel's T: a case study, J. Symbolic Logic, to appear.