Continuous Normalization for the Lambda-Calculus and Gödel's T

Klaus Aehlig[⋆] and Felix Joachimski

{aehlig|joachski}@mathematik.uni-muenchen.de
Mathematisches Institut, Ludwig-Maximilians-Universität München
Theresienstrasse 39, 80333 München, Germany

Abstract. Building on previous work by Mints, Buchholz and Schwichtenberg, a simplified version of continuous normalization for the untyped λ-calculus and Gödel's T is presented and analyzed in the coalgebraic framework of non-wellfounded terms with so-called repetition constructors.
The primitive recursive normalization function is uniformly continuous w.r.t. the natural metric on non-wellfounded terms. Furthermore, the number of necessary repetition constructors is locally related to the number of reduction steps needed to reach and the size of the normal form (as represented by the Böhm tree).
It is also shown how continuous normal forms relate to derivations of strong normalizability in the typed λ-calculus and how this leads to new bounds for the sum of the height of the reduction tree and the size of the normal form.
Finally, the methods are extended to an infinitary λ-calculus with ω-rule and permutative conversions and this is used to derive a strong form of normalization for an iterative version of Gödel's system T, leading to a value table semantics for number-theoretic functions.

Introduction

Continuous normalization has been introduced by Mints [Min78,KMS75] in order to separate cut-elimination for semiformal systems from their ordinal analysis. The operational aspects of normalization, i.e., the manipulations on infinitary derivations, are isolated and described independently of the system's proof theoretic complexity, but at the expense of introducing the void logical rule of *repetition* to balance derivation trees:

$$(\mathcal{R})\ \frac{\Gamma \vdash A}{\Gamma \vdash A}$$

Note that this rule is both logically valid and preserves the subformula property.

Using $(\mathcal{R})$, the cut-elimination operator becomes a primitive recursive function and can be studied in its own right. As remarked by Mints it can even be

⋆ Supported by the "Graduiertenkolleg Logik in der Informatik" of the Deutsche Forschungsgemeinschaft

applied to non-wellfounded derivations, because the underlying manipulations are local, or more precisely, *continuous* w.r.t. the standard metric on infinitary trees: the normalization procedure requires only about as much information of the input as it produces output, using ($\mathcal{R}$) as the last inference rule of the normal derivation, if the result cannot immediately be determined ("please wait").

The concept of continuous normalization has been adapted from sequent calculi to a natural deduction setting by Schwichtenberg [Sch98], using Buchholz's notation systems for infinitary derivations [Buc91].

Continuous normalization for Λ. Along similar lines of thought, a simplified version of continuous normalization for the untyped λ-calculus has been defined earlier [AJ02]. Since normalization need not terminate on arbitrary λ-terms and may even lead to infinite normal forms, this seems to be a natural setting for the study of continuous normalization. Instead of partly defined trees, (not necessarily well-founded) normal forms in an extended language with $\mathcal{R}$. While for instance the head constructor of the normal form of λxr must be λ, the head constructor of an application rs depends on whether r is an abstraction, a variable, or again an application. The continuous normalization function therefore outputs $\mathcal{R}$, before further analysing r to find out whether s will be used for a substitution (if r were an abstraction) or has to be processed next (if r is a variable). As a consequence, the result of the diverging term $(\lambda x.xx)\lambda x.xx$ is an infinite sequence of repetition rules.

The coinductive λ-calculus Λ^{co} arises by a coinductive interpretation of the grammar of the wellfounded λ-calculus [Joa01] and provides a sound framework to model non-wellfounded λ-terms and Böhm trees [Bar77]. To dispense with the intricate mechanism of bound variable renaming in semiformal term systems, it is based on a deBruijn-style management of bound variables [Bru72]. Apart from all terms of the usual λ-calculus, Λ^{co} harbors interesting objects like the directly defined (and even well-typed) fixed point $Y_r := r(Y_r)$ of an arbitrary term r.

Counting $\mathcal{R}$s. As shown elsewhere [AJ02] (and briefly reviewed here), the number of repetition constructors in the continuous normal form r^β of a term r can be precisely related to the number of reduction steps and the "size" of the resulting Böhm tree. This analysis is achieved by means of an additional variant β of the repetition constructor $\mathcal{R}$, which —as the name insinuates— corresponds to a reduction step in the standard reduction sequence. For example,[1] normalization of the fixpoint combinator $Y := \lambda f.(\lambda x.f(xx))\lambda x.f(xx)$ applied to $K := \lambda y\lambda xy$

[1] For the convenience of the reader we use named λ-terms as obvious abbreviations for the corresponding de Bruijn-terms in these introductory examples.

results in

$$(YK)^\beta = \quad \mathcal{R}\beta\ \mathcal{R}\beta\mathcal{R}\beta\lambda x. \qquad \mathcal{R}\beta\mathcal{R}\beta\lambda x. \qquad \mathcal{R}\beta\mathcal{R}\beta\lambda x. \qquad \mathcal{R}\beta\mathcal{R}\beta\lambda x. \ldots$$

while the variant $\hat{Y} := \lambda f.(\lambda x.f(f(xx)))\lambda x.f(f(xx))$ yields

$$(\hat{Y}K)^\beta = \quad \mathcal{R}\beta\ \mathcal{R}\beta\mathcal{R}\beta\lambda x. \qquad \quad \mathcal{R}\beta\lambda x. \qquad \mathcal{R}\beta\mathcal{R}\beta\lambda x. \qquad \quad \mathcal{R}\beta\lambda x. \ldots$$

and for the fixed point combinator $\Theta := (\lambda x, f.f(xxf))\lambda x, f.f(xxf)$ we obtain

$$(\Theta K)^\beta = \mathcal{R}\mathcal{R}\beta \quad \beta\mathcal{R}\beta\lambda x.\ \mathcal{R}\mathcal{R}\beta\beta\mathcal{R}\beta\lambda x.\ \mathcal{R}\mathcal{R}\beta\beta\mathcal{R}\beta\lambda x.\ \mathcal{R}\mathcal{R}\beta\beta\mathcal{R}\beta\lambda x. \ldots$$

So, apart from computing the non-wellfounded normal form $\lambda x\lambda x\lambda x\ldots$,[2] the result yields insight into the normalization behaviour of its argument and thus permits a more detailed analysis: It shows that unfolding ΘK requires 3 reductions

$$\begin{aligned} \Theta K &= (\lambda x, f.f(xxf))\lambda x, f.f(xxf))K \\ &\to (\lambda f.f(\Theta f))K \\ &\to K(\Theta K) \\ &\to \lambda x.\Theta K \end{aligned}$$

to obtain the next λ, while ${}_KY := (\lambda x.K(xx))\lambda x.K(xx)$ only needs two:

$$(\lambda x.K(xx))\lambda x.K(xx) \to {}_KY \to \lambda x\ {}_KY$$

and one unfolding of ${}_K\hat{Y} := (\lambda x.K(K(xx)))\lambda x.K(K(xx))$ computes the second approximation of the fixed point of K

$$\begin{aligned} {}_K\hat{Y} &= (\lambda x.K(K(xx)))\lambda x.K(K(xx)) \\ &\to K(K\ {}_K\hat{Y}) \\ &\to \lambda z(K\ {}_K\hat{Y}) \\ &\to \lambda z\lambda z\ {}_K\hat{Y}. \end{aligned}$$

Not very surprising, the directly defined fixpoint Y_K^{co} is the most efficient: $Y_K^{\mathrm{co}} = KY_K^{\mathrm{co}} \to \lambda x Y_K^{\mathrm{co}}$. In fact, the continuous normal forms of the various fixed point combinators used so far are

$$\begin{aligned} \Theta^\beta &= \mathcal{R}\beta\lambda x.\mathcal{R}(x(\mathcal{R}\mathcal{R}\beta\beta\mathcal{R}(x(\mathcal{R}\mathcal{R}\beta\beta\mathcal{R}(x(\mathcal{R}\mathcal{R}\beta\beta\mathcal{R}(x(\mathcal{R}\mathcal{R}\beta\beta\mathcal{R}(x(\ldots \\ Y^\beta &= \lambda x.\mathcal{R}\beta\mathcal{R}(x(\ \ \mathcal{R}\beta\ \ \mathcal{R}(x(\ \ \mathcal{R}\beta\ \ \mathcal{R}(x(\ \ \mathcal{R}\beta\ \ \mathcal{R}(x(\ \ \mathcal{R}\beta\ \ \mathcal{R}(x(\ldots \\ \hat{Y}^\beta &= \lambda x.\mathcal{R}\beta\mathcal{R}(x(\qquad \mathcal{R}(x(\ \ \mathcal{R}\beta\ \ \mathcal{R}(x(\qquad \mathcal{R}(x(\ \ \mathcal{R}\beta\ \ \mathcal{R}(x(\ldots \\ (Y_x^{\mathrm{co}})^\beta &= \qquad \mathcal{R}(x(\qquad \mathcal{R}(x(\qquad \mathcal{R}(x(\qquad \mathcal{R}(x(\qquad \mathcal{R}(x(\ldots \end{aligned}$$

Cut-elimination. The classical proof-theoretic approach to normalization in the λ-calculus stepwise reduces cuts (i.e., applications) of maximal rank to cuts

[2] Note that all three terms have undefined Böhm trees.

of lower rank, thus progressively eliminating β-redexes until a normal form is reached. In contrast, the continuous normalization function computes the normal form of a term in one pass, with some possible repetition rules interjected.

It is all the more surprising that the cut-elimination process actually constructs a reduction sequence that exactly corresponds to the reduction strategy underlying the continuous normalization function. Even more, the continuous normal form with its repetition constructors serves as a denotation of this reduction sequence.

Bounds. This correspondence can be exploited to derive bounds for the length of reduction sequences in the typed λ-calculus, by using size annotations to a cut-elimination based strong normalization proof. Strengthening known results of Beckmann [Bec01], these bounds limit the sum of the size of the resulting normal form *and* the height of the reduction tree of a term r by the expression $2_{\mathrm{rk}\ r}||r||$, where rk r is the level of the greatest type occurring in r and $||r||$ is its size.

Adding the ω-rule. We retrace the traditional Schütte-style [Sch51] proof-theoretic analysis of Peano arithmetic by embedding a λ-calculus based formulation of Gödel's system T [Göd58] into the semiformal system T_∞ with an infinitary branching ω-rule, to which we can extend both the continuous normalization function and the cut-elimination proof.

In contrast to conventional treatments of T_∞, this proof allows for permutative conversions and normalization of open terms rather than only numerals. This leads to particularly simple normal forms, which for the instance of number-theoretic functions serves to construct a value table.

In order to illustrate the correspondence to ordinal-based normalization proofs for T, we also reprove the classical bound ε_0 for its proof-theoretic strength.

Outline of the contents. Section 1 recalls the definition of the coinductive λ-calculus and the basic concepts of lifting, substitution, reduction and normal forms. Section 2 briefly reviews the results of continuity and soundness for continuous normalization [AJ02]. Section 3 deviates from loc.cit. in its definition of well-formedness for $\mathcal{R}$-terms, using it to provide a structural analysis of continuous normal forms and elaborates the connection between continuous normalization and leftmost-outermost reduction. Section 4 shows how continuous normal forms serve to denote SN-derivations and uses this connection to establish new bounds for the sum of the reduction tree height and the normal form as a function of the size and the rank of a term. En passant the relationship between cut-elimination and continuous normalization is clarified. Section 5 applies the method of the previous section to system T_∞, a λ-calculus with the infinitary ω-rule. Section 6 embeds Gödel's system T into T_∞, deriving normalization as a corollary to the previous section.

1 The Coinductive λ-Calculus with $\mathcal{R}$ and β

The coinductive λ-calculus Λ^{co} arises by a coinductive interpretation of the defining grammar of the usual λ-calculus. Since this construction includes infinitary λ-terms, a variable's property of being new w.r.t. a given term is no longer decidable. Terms like r_0 with $r_n := x_n r_{n+1}$ (where $x_0, x_1, x_2, \ldots$ is an enumeration of all variables) may even contain all variables. It is thus reasonable to retreat into a de Bruijn discipline [Bru72] in handling free and bound variables in order to keep constructions like substitution primitive recursive and forgo the tedious problems of α-equality on infinitary terms.

1.1. Terms. Terms of the inductive and coinductive λ-calculus and their extensions by $\mathcal{R}$ and β are given by[3]

$$\begin{array}{lll}
\Lambda & \ni r, s ::= & k \mid rs \mid \lambda r \\
\Lambda^{\mathrm{co}} & \ni r, s ::=^{\mathrm{co}} & k \mid rs \mid \lambda r \\
\Lambda_{\mathcal{R}} & \ni r, s ::= & k \mid rs \mid \lambda r \mid \mathcal{R}r \mid \beta r \\
\Lambda^{\mathrm{co}}_{\mathcal{R}} & \ni r, s ::=^{\mathrm{co}} & k \mid rs \mid \lambda r \mid \mathcal{R}r \mid \beta r
\end{array}$$

Notation. x, y, k, l, m, n range over natural numbers. Obviously $\Lambda^{\mathrm{co}}_{\mathcal{R}} \supset \Lambda^{\mathrm{co}} \supset \Lambda \subset \Lambda_{\mathcal{R}} \subset \Lambda^{\mathrm{co}}_{\mathcal{R}}$. The notorious dot notation is applied as follows: A dot stands for a pair of parentheses that open at the dot and close as far right as syntactically possible. For instance $\lambda\lambda\lambda.2\,0.1\,0$ stands for $\lambda\lambda\lambda((2\,0)(1\,0))$, i.e., the combinator S. $\vec{r}^{\,n}$ (the superscript n will be omitted whenever reasonable) denotes a possibly empty list of terms $r_1, \ldots, r_n$. ε stands for the empty list. A comma is used for pre- and postfixing terms as well as appending lists.

Examples. At the term root, the variable number n corresponds to the n-th variable in a fixed enumeration. λ abstracts the variable 0, so that $\lambda 0$ corresponds to λxx in a named setting. If xy is represented[4] by $4\,2$, the term $\lambda z.xy$ reads $\lambda.5\,3$. The term $\lambda x\lambda y.xy$ is represented by $\lambda\lambda.1\,0$.

By the guarded[5] recursive definition $Y^{\mathrm{co}}_r := rY^{\mathrm{co}}_r$ a direct implementation of a fixpoint for any term r is admissible in Λ^{co}. Less reasonable terms are $r := rr$ or $r := \lambda r$. In $\Lambda^{\mathrm{co}}_{\mathcal{R}}$ we may define

$$\bot_n ::=^{\mathrm{co}} \mathcal{R}\bot_{n+1} \mid \beta\bot_{n-1} \mid \lambda\bot_n.$$

[3] More formally, the coinductive λ-calculus arises as the carrier of the final coalgebra for the Set-endofunctor $LX := \mathbb{N} + X^2 + X$, which (like final coalgebras of all polynomial functors) has a pleasingly simple construction and is equipped with an isomorphism $\Lambda^{\mathrm{co}} \longleftrightarrow L(\Lambda^{\mathrm{co}})$ by Lambek's theorem [Rut00]. The arrow $L(\Lambda^{\mathrm{co}}) \longrightarrow \Lambda^{\mathrm{co}}$ is given by the constructors of the λ-calculus (variables, application, abstraction). The notation $::=^{\mathrm{co}}$ signifies that this coinductive interpretation of the grammar is invoked.

[4] For the sake of readability, we will only use variable numbers < 10 in examples.

[5] The concept of guarded recursion has been proposed by Coquand [Coq94] and further developed by many others [Gim95, TT97] as a method to define non-wellfounded objects that is reducible to corecursion. It allows quite liberal recursive calls to the defined function, as long as they occur inside the scope of a constructor ("guard").

It can be shown that each $\bot_0$ arises exactly as a normal form of a term with undefined Böhm tree [AJ02].

1.2. Observational equality. Define the equivalence relation $\simeq_k$ on $\Lambda^{\mathrm{co}}_{\mathcal{R}}$ inductively by

$$\frac{}{r \simeq_0 r'} \qquad \frac{}{x \simeq_k x} \qquad \frac{r, s \simeq_k r', s'}{rs \simeq_{k+1} r's'} \qquad \frac{r \simeq_k r'}{\lambda r, \mathcal{R}r, \beta r \simeq_{k+1} \lambda r', \mathcal{R}r', \beta r'}$$

Here we used the abbreviation $\vec{r}^{\,n} \simeq_k \vec{s}^{\,n} \;:\Leftrightarrow\; r_1 \simeq_k s_1 \;\wedge\; \ldots \;\wedge r_n \simeq_k s_n$.

Remarks. Obviously $r \simeq_{k+n} r'$ implies $r \simeq_k r'$ (weakening). $\simeq_k$-equivalence classes define the open sets of a topology on terms in coinductive calculi which is also induced by the metric $\mathrm{d}(r, s) := \frac{1}{1+\sup_k r \simeq_k s}$.

Equality on non-wellfounded terms is given by the bisimulation

$$r = s \;:\Longleftrightarrow\; \forall k. r \simeq_k s.$$

1.3. Lifting. The canonical way to find a new free variable in a term is to lift all variable numbers by 1, so that 0 becomes new.

More precisely, we define by guarded recursion $(-)\!\uparrow_n : \Lambda^{\mathrm{co}}_{\mathcal{R}} \longrightarrow \Lambda^{\mathrm{co}}_{\mathcal{R}}$.

$$\begin{array}{rlrl}
(rs)\!\uparrow_n &:= r\!\uparrow_n s\!\uparrow_n & (\lambda r)\!\uparrow_n &:= \lambda . r\!\uparrow_{n+1} \\
(\mathcal{R}r)\!\uparrow_n &:= \mathcal{R}. r\!\uparrow_n & (\beta r)\!\uparrow_n &:= \beta . r\!\uparrow_n \\
k\!\uparrow_n &:= \begin{cases} k & \text{if } k < n \\ k+1 & \text{otherwise} \end{cases} & r\!\uparrow &:= r\!\uparrow_0
\end{array}$$

For instance $(1\,3)\!\uparrow_2 = 1\,4$ and $(\lambda.0\,1\,3)\!\uparrow_0 = \lambda.0\,2\,4$.

Remark. Since lifting only affects variables, it is continuous with the identity as modulus of continuity, i.e., $r \simeq_k s$ implies $r\!\uparrow_l \simeq_k s\!\uparrow_l$.

1.4. Substitution. Define by guarded recursion $-[-] : \Lambda^{\mathrm{co}}_{\mathcal{R}} \times \Lambda^{\mathrm{co}}_{\mathcal{R}} \longrightarrow \Lambda^{\mathrm{co}}_{\mathcal{R}}$.

$$\begin{array}{rlrl}
(rs)[t]_n &:= r[t]_n\, s[t]_n & (\lambda r)[t]_n &:= \lambda . r[t\!\uparrow]_{n+1} \\
(\mathcal{R}r)[t]_n &:= \mathcal{R}. r[t]_n & (\beta r)[t]_n &:= \beta . r[t]_n \\
k[t]_n &:= \begin{cases} k & \text{if } k < n \\ t & \text{if } k = n \\ k-1 & \text{otherwise,} \end{cases} & r[t] &:= r[t]_0
\end{array}$$

This notion of substitution is tailored for β-reduction only, so that for instance the identity axiom $\exists \theta . r\theta = r$ does not hold. A general substitution for non-wellfounded term systems which satisfies the usual monadic laws can be found in [FPT99]. For an adaptation to Λ^{co} see [Joa01].

It is straightforward to verify that substitution is again continuous with identity as modulus of continuity:

Proposition 1. $r, s \simeq_k r', s' \implies r[s]_l \simeq_k r'[s']_l$.

The following lemma states the usual commutation properties for substitution, recast in the terminology of de Bruijn-terms.

Lemma 2.

(1) $r{\uparrow_m}{\uparrow_{m+n+1}} = r{\uparrow_{m+n}}{\uparrow_m}$.
(2) $r[s]_{m+n}{\uparrow_n} = r{\uparrow_n}[s{\uparrow_n}]_{m+n+1}$
(3) $r{\uparrow_{m+n+1}}[s{\uparrow_{m+n}}]_m = r[s]_m{\uparrow_{m+n}}$
(4) $r{\uparrow_m}[s]_m = r$.
(5) $r[s]_m[t]_{m+n} = r[t{\uparrow_m}]_{m+n+1}[s[t]_{m+n}]_m$.

Proof. Guarded induction.[6] □

1.5. β-reduction. The reduction relation $\to$ is only needed and defined on the inductive calculus Λ. It is the compatible closure of elementary β-reduction $(\lambda r)s \mapsto r[s]$, i.e., defined inductively by

$$\frac{}{(\lambda r)s \to r[s]} \qquad \frac{r \to r'}{\lambda r, rs, sr \to \lambda r', r's, sr'}$$

(with pointwise reading of $\vec{r} \to \vec{s}$). $\to^*$ is the reflexive transitive closure of $\to$.

Remark. If used in Λ^{co}, the reduction $\to$ is no longer confluent: With $r := 0r$ and $s := ((\lambda 0)0)s$ we have

$$r \;{}^*\!\not\leftarrow\; s \leftarrow (\lambda r)((\lambda 0)0) \to (\lambda r)0 \to r \not\to^* s.$$

For a more precise analysis and positive confluence results see [Joa03].

1.6. Normal forms. The set of *normal forms*

$$\mathrm{NF} \ni r, s ::=^{\mathrm{co}} x\vec{r} \mid \lambda r \mid \mathcal{R}r \mid \beta r$$

contains only terms without redexes. Note that $\mathrm{NF} \cap \Lambda$ is the usual set of normal forms of Λ. However, there are some normal terms in $\Lambda^{\mathrm{co}}_{\mathcal{R}}$ which are not captured by this cogrammar, such as $(\mathcal{R}0)0$.[7]

[6] Guarded induction is the corecursive construction of a non-wellfounded proof by means of guarded recursion. By the bisimulation theorem [Rut00], $\Lambda^{\mathrm{co}} \ni r = s \iff \forall k.r \simeq_k s$, because $\bigcap_k \simeq_k$ is a bisimulation. Thus guarded induction in a proof of $r = s$ corresponds to the proof of $\forall k.r \simeq_k s$ by induction on k in the same way as guarded recursion *is* recursion on the depth of observations.

[7] The cogrammar for *all* terms without redexes is $r, s ::=^{\mathrm{co}} x\vec{r} \mid \lambda r \mid (\mathcal{R}r)\vec{s} \mid (\beta r)\vec{s}$.

2 Continuous Normalization

This section defines the primitive recursive normalization function $(-)^\beta$. The result of r^β can be understood as the normal form of r, enriched by information on the reduction sequence that was used to reach it.

The definition of r^β takes recourse to an auxiliary function $r@\vec{s}$, which intuitively should compute the normal form of $r\vec{s}$.

2.1. Definition. We define $r@\vec{s} \in \mathrm{NF}$ (with $r, \vec{s} \in \Lambda^{\mathrm{co}}_{\mathcal{R}}$) by guarded recursion, using the abbreviation $r^\beta := r@\varepsilon$.

$$\begin{array}{rl} (\lambda r)@(s,\vec{s}) := \beta.r[s]@\vec{s} & \qquad (\lambda r)@\varepsilon := \lambda r^\beta \\ x@\vec{s} := x\vec{s}^{\,\beta} & \qquad (\mathcal{R}r)@\vec{s} := \mathcal{R}.r@\vec{s} \\ (rs)@\vec{s} := \mathcal{R}.r@(s,\vec{s}) & \qquad (\beta r)@\vec{s} := \beta.r@\vec{s} \end{array}$$

The normalization function outputs $\mathcal{R}$ whenever it faces an application rs, because it cannot foresee what to do with the argument s. When it next encounters an abstraction $r = \lambda r'$, the s will be used for the substitution $r'[s]$, so the $\mathcal{R}$ is justified ex post. If on the other hand a variable $r = x$ should follow then the s will be further normalized to produce the normal form of xs. The $\mathcal{R}$ produced in the step $(rs)^\beta = \mathcal{R}.r@s$ thus accounts for the application in the normal form xs^β.

Example. $(SKK)^\beta = \mathcal{R}\mathcal{R}\beta\beta\lambda\mathcal{R}\mathcal{R}\beta\beta 0$ with $S := \lambda\lambda\lambda.2\,0.1\,0$, $K := \lambda\lambda 1$. Note that $SKK \to^2 \lambda.K0.K0 \to^2 \lambda 0$.

Remarks.

- It is easy to see that the β-constructor is not necessary to ensure well-definedness, but it guarantees that the modulus of continuity is the identity.
- For $r \in \Lambda$, the last two clauses are not needed to compute r^β.

2.2. Continuity. The following lemma establishes that $(-)^\beta$ is continuous with modulus of continuity $k \mapsto k$.

Lemma 3. $r \simeq_k r' \ \wedge\ \vec{s} \simeq_k \vec{s'} \implies r@\vec{s} \simeq_k r'@\vec{s'}$.

Proof. Induction on k, using continuity of substitution. □

2.3. Normalization. The next goal is to verify that the result computed by r^β is actually the normal form of r, if it is finite. Since r^β might contain some $\mathcal{R}$ and β, we have to eliminate them to prove this correctness property. To this end we define $r^* \in \Lambda$ by recursion on $r \in \Lambda_{\mathcal{R}}$:

$$x^* := x, \qquad (rs)^* := r^* s^*, \qquad (\lambda r)^* := \lambda r^*, \qquad (\mathcal{R}r)^* := (\beta r)^* := r^*.$$

Proposition 4. $r \in \Lambda \ \wedge\ r^\beta \in \Lambda_{\mathcal{R}} \implies r \to^* r^{\beta *}$.

Proof. Induction on the *size* of r^β, distinguishing cases as to the form of r. □

3 Well-formedness

This section introduces the coinductive concept of well-formedness, the inductive counterpart of which will turn out to correspond to derivations of normalizability in the next section.

3.1. Definition. Informally, well-formedness of a term r requires that all $\mathcal{R}$s occuring in r are either justified by a following β or correspond to one element of a variable application $x\vec{r}$.

The concept will be defined by means of a coinductive calculus that derives judgements of the form $r \vdash s \,\square\, \vec{t}$, to be read "$r$ is well-formed w.r.t. $s, \vec{t}$" with $r \in \Lambda^{\mathrm{co}}_{\mathcal{R}}$ and $s, \vec{t} \in \Lambda^{\mathrm{co}}$. $t \vdash r$ abbreviates $t \vdash r \,\square\, \varepsilon$ and $\vdash r$ expresses the existence of a t such that $t \vdash r$.

$$\frac{t \vdash r \,\square\, s, \vec{s}}{\mathcal{R}t \vdash rs \,\square\, \vec{s}} \qquad \frac{\vec{t} \vdash \vec{r}}{x\vec{t} \vdash x \,\square\, \vec{r}} \qquad \frac{t \vdash r}{\lambda t \vdash \lambda r} \qquad \frac{t \vdash r[s] \,\square\, \vec{s}}{\beta t \vdash \lambda r \,\square\, s, \vec{s}}$$

Proposition 5. *$r@\vec{s} \vdash r \,\square\, \vec{s}$.*

3.2. Bounds for $\mathcal{R}$. The first goal is to show that each $\mathcal{R}$ produced during continuous normalization corresponds to either a β-reduction or an application in the normal form. In order to prove this for possibly non-wellfounded normal forms, we need the concept of paths.

Definition (Paths). A *path* is a list of natural numbers, i.e., $\zeta ::= \varepsilon \mid k \cdot \zeta$. The set of paths of a term r is given inductively by

$$\frac{}{\varepsilon \in r} \qquad \frac{\zeta \in r_k}{k \cdot \zeta \in x\vec{r}} \qquad \frac{\zeta \in r}{k \cdot \zeta \in \lambda r} \qquad \frac{\zeta \in r}{k \cdot \zeta \in \mathcal{R}r} \qquad \frac{\zeta \in r}{k \cdot \zeta \in \beta r}$$

ζ *is complete in* r (written $\zeta \in_c r$) iff $\varepsilon \in r$ is used only for abstractions λs and variable eliminations $x\vec{r}$. In other words, at the end of a complete path there are no pending terms to be applied.

(Only) for valid paths $\zeta \in r$ we define the number $\mathcal{R}_\zeta r$ of $\mathcal{R}$s, the number $\beta_\zeta r$ of βs and the number $A_\zeta r$ of applications in the path by

$$\begin{array}{llllll}
\mathcal{R}_{k\cdot\zeta}\lambda r & := \mathcal{R}_\zeta r & \beta_{k\cdot\zeta}\lambda r & := \beta_\zeta r & A_{k\cdot\zeta}\lambda r & := A_\zeta r \\
\mathcal{R}_{k\cdot\zeta}\mathcal{R}r & := 1{+}\mathcal{R}_\zeta r & \beta_{k\cdot\zeta}\mathcal{R}r & := \beta_\zeta r & A_{k\cdot\zeta}\mathcal{R}r & := A_\zeta r \\
\mathcal{R}_{k\cdot\zeta}\beta r & := \mathcal{R}_\zeta r & \beta_{k\cdot\zeta}\beta r & := 1{+}\beta_\zeta r & A_{k\cdot\zeta}\mathcal{R}r & := A_\zeta r \\
\mathcal{R}_{k\cdot\zeta}(x\vec{r}) & := \mathcal{R}_\zeta r_k & \beta_{k\cdot\zeta}(x\vec{r}) & := \beta_\zeta r_k & A_{k\cdot\zeta}(x\vec{r}^{\,n}) & := n{+}A_\zeta r_k \\
\mathcal{R}_\varepsilon r & := 0 & \beta_\varepsilon r & := 0 & A_\varepsilon(x\vec{r}^{\,n}) & := n \\
 & & & & A_\varepsilon r & := 0 \text{ otherwise}
\end{array}$$

Lemma 6. *$s \vdash r \,\square\, \vec{r}^{\,n} \implies n + \mathcal{R}_\zeta s \geq \beta_\zeta s + A_\zeta s$ with equality for $\zeta \in_c s$.*

Proof. Induction on $\zeta \in s$.
Case ε. If s is not of the form $x\vec{r}$ then the claim is trivial, because $n + \mathcal{R}_\varepsilon s = n \geq 0 = \beta_\varepsilon s + A_\varepsilon s$. For a variable elimination $x\vec{r}^{\,n} \vdash x \circ \vec{t}$ we compute

$$n + \mathcal{R}_\varepsilon(x\vec{r}) = n + 0 = n = \beta_\varepsilon(x\vec{r}) + A_\varepsilon(x\vec{r}).$$

Case $l \cdot \zeta$, subcase $\mathcal{R}r \vdash st \circ \vec{t}$ from $r \vdash s \circ t, \vec{t}$.

$$\begin{aligned} n + \mathcal{R}_{l\cdot\zeta}\mathcal{R}r &= n + 1 + \mathcal{R}_\zeta r \\ &\geq \beta_\zeta r + A_\zeta r && \text{by IH} \\ &= \beta_{l\cdot\zeta}\mathcal{R}r + A_{l\cdot\zeta}\mathcal{R}r. \end{aligned}$$

Subcase $x\vec{r}^{\,n} \vdash x \circ \vec{t}$ from $\vec{r} \vdash \vec{t}$.

$$\begin{aligned} n + \mathcal{R}_{l\cdot\zeta}(x\vec{r}) &= n + \mathcal{R}_\zeta r_l \\ &\geq n + \beta_\zeta r_l + A_\zeta r_l && \text{by IH} \\ &= \beta_{l\cdot\zeta}(x\vec{r}) + A_{l\cdot\zeta}(x\vec{r}^{\,n}). \end{aligned}$$

Subcase $\beta r \vdash \lambda r \circ s, \vec{s}^{\,n}$ from $r \vdash r[s] \circ \vec{s}$.

$$\begin{aligned} n{+}1 + \mathcal{R}_{l\cdot\zeta}\beta r &= 1 + n + \mathcal{R}_\zeta r \\ &\geq 1 + \beta_\zeta r + A_\zeta r && \text{by IH} \\ &= \beta_{l\cdot\zeta}\beta r + A_{l\cdot\zeta}\beta r. \end{aligned}$$

The remaining case of a λ-abstraction is a simple application of the induction hypothesis. $\square$

As an instance of this lemma we obtain for well-formed s that the number of $\mathcal{R}$s on each complete path is precisely the number of βs plus the number of applications.

3.3. Leftmost-outermost reduction. Our next goal is to establish a formal connection between the occurences of the constructor β and β-reductions of the term under consideration. Following [JM00] we use an inductive characterization of leftmost-outermost reduction strategies: $r \rightsquigarrow_n s$ with $r, s \in \Lambda^{\mathrm{co}}$ ("r standard reduces to s in n steps) is given inductively by the following rules

$$(v)\ \frac{\vec{r} \rightsquigarrow_{\vec{n}} \vec{s}}{x\vec{r} \rightsquigarrow_{\Sigma\vec{n}} x\vec{s}} \qquad (\lambda)\ \frac{r \rightsquigarrow_n s}{\lambda r \rightsquigarrow_n \lambda s} \qquad (\beta)\ \frac{r[s]\vec{s} \rightsquigarrow_n t}{(\lambda r)s\vec{s} \rightsquigarrow_{n+1} t} \qquad (r)\ \frac{}{r \rightsquigarrow_0 r}$$

($\vec{r} \rightsquigarrow_{\vec{n}} \vec{s}$ is read pointwise)

By the standardization theorem, $r \to^* s \in \mathrm{NF} \cap \Lambda$ implies that there exists an n with $r \rightsquigarrow_n s$.

3.4. $\mathcal{R}$-elimination. Let $\rhd_\mathcal{R}$ and $\rhd_\beta$ be the compatible closures of $\mathcal{R}r \rhd_\mathcal{R} r$ and $\beta r \rhd_\beta r$. $\rhd_\mathcal{R}^n$ stands for n steps of $\rhd_\mathcal{R}$. Let $\rhd_n^k$ contain reduction sequences with k $\rhd_\mathcal{R}$ and n $\rhd_\beta$-reductions (mixed ad libitum). Furthermore, we set $\rhd_n := \rhd_n^n$.

Lemma 7. $r \rightsquigarrow_n s \implies r^\beta \rhd_n s^\beta$.

Proof. Induction on $\rightsquigarrow_n$. $\square$

3.5. Weakly normalizing terms. If a term r actually has a finite normal form then this is unique and will be denoted by nf r. In this case we can slightly strengthen the last lemma.

Definition (Application count). For $r \in \Lambda \cap \mathrm{NF}$ we define recursively

$$|x\vec{r}^{\,n}|^a := n + \Sigma\, |\vec{r}\,|^a, \qquad |\lambda r|^a := |r|^a.$$

Proposition 8. $r \in \Lambda \cap \mathrm{NF} \implies r^\beta \rhd_{\mathcal{R}}^{|r|^a} r.$

Corollary 9. $r \rightsquigarrow_n s \in \mathrm{NF} \cap \Lambda \implies r^\beta \rhd_n^{n+|s|^a} s.$

Hence, for every weakly normalizing term r, the normal form is computed by r^β and we have precise information on the number of steps in the standard reduction sequence leading to it.

4 Bounds for $\vec{\Lambda}$

In this section we relate our continuous normalization function to traditional cut-elimination procedures, thus retracing the traditional proof-theoretic steps from continuous cut-elimination to the (ordinal) analysis of derivation heights. It turns out that the $\mathcal{R}, \beta$-annotated normal form of r as computed by r^β denotes a derivation that r is strongly normalizing. For the case of the simply-typed λ-calculus this will be used to derive bounds for the height of the reduction tree *and* the resulting normal form, thus improving results by Schwichtenberg and Beckmann [Sch98, Bec01].

4.1. Perpetual extension. As it was stated in section 2 the normalization function $(-)^\beta$ computes the normal form using the leftmost-outermost strategy. This can be used to derive bounds for the length of reduction sequences that follow this strategy. Yet by only a minor tweak to the definition of $(-)^\beta$ it is possible to make the reduction strategy *perpetual* (in the sense of [vRSSX99]), so that the number of steps performed limits the height of the reduction tree altogether.

To this end we slightly modify the grammar of $\Lambda_{\mathcal{R}}$ (and its coinductive analogue) by adding a term as an index of the β-constructor:

$$\Lambda_{\mathcal{R}}^{(\mathrm{co})} \ni r, s ::=^{(\mathrm{co})} k \ \mid\ rs \ \mid\ \lambda r \ \mid\ \mathcal{R}r \ \mid\ \beta_s r.$$

The respective clause in the definition of $(-)^\beta$ is altered as follows:

$$(\lambda r)@(s, \vec{s}\,) := \beta_{s^\beta}.r[s]@\vec{s}.$$

Remark. Although it will not be further elaborated, the relationship to perpetual reduction strategies is seen as follows: modify the rule (β) of the definition of $\rightsquigarrow_n$ in subsection 3.3 to include reductions on the side term:

$$(\beta)\ \frac{r[s]\vec{s} \rightsquigarrow_n t \qquad s \rightsquigarrow_m s' \qquad 0 \notin \mathrm{FV}\, r}{(\lambda r)s\vec{s} \rightsquigarrow_{n+m+1} t} \qquad (\beta')\ \frac{r[s]\vec{s} \rightsquigarrow_n t \qquad 0 \in \mathrm{FV}\, r}{(\lambda r)s\vec{s} \rightsquigarrow_{n+1} t}$$

Herein, the free variables $\mathrm{FV}\, r$ are defined recursively as usual. The strategy $\rightsquigarrow := \bigcup_n \rightsquigarrow_n$ is perpetual: it finds the longest possible reduction strategy (as reproved in [vRSSX99] with different notations).

4.2. Types. *(Simple) Types* ρ, σ, τ are generated from basic types ι by $\rho \to \sigma$. The *level* of ρ is given by $\mathrm{lev}\ \iota := 0$ and $\mathrm{lev}\ (\rho \to \sigma) := \max\{1 + \mathrm{lev}\ \rho, \mathrm{lev}\ \sigma\}$.

The typed λ-calculi (in Church form) are obtained from the untyped ones by attaching types to λ-abstractions: $\lambda^\rho r$. Relative to a sequence $\phi = (\rho_n)_{n\in\mathbb{N}}$ of types, we can then determine the typable terms and their type by the following rules:

$$\frac{\phi \Vdash r : \rho \to \sigma \qquad \phi \Vdash s : \rho}{\phi \Vdash rs : \sigma} \qquad \frac{\rho, \phi \ \Vdash r : \sigma}{\phi \ \Vdash \lambda^\rho r : \rho \to \sigma}$$

Note that these rules have to be interpreted coinductively for terms of Λ^{co}.

As usual, we have subject reduction: $r \to s \ \&\ \phi \Vdash r : \rho \implies \phi \Vdash s : \rho$.

Assuming a fixed ϕ, we write $r : \rho$ for $\phi \Vdash r : \rho$. We omit the type annotation of λ-abstractions wherever possible. Also we will decorate (sub-)terms with types in superscripts (as in $r^\rho s$) in order to signify that they are typable and get the respective type relative to ϕ. Suppressing the erasure of type annotations, $\vec{\Lambda}^{(\mathrm{co})}$ denotes the set of typable terms in $\Lambda^{(\mathrm{co})}$.

For the rest of this section we restrict our focus (and all quantifiers) to typable terms.

4.3. The sets NF_k. To model cut-elimination we introduce the following extensions of the normal forms of $\Lambda^{(\mathrm{co})}_{\mathcal{R}}$:

$$\mathrm{NF}_k^{(\mathrm{co})} \ni r, s ::=^{(\mathrm{co})} x\vec{r} \mid \lambda r \mid \mathcal{R}r \mid \beta_s r \mid \mathcal{C}(r^{\rho\to\sigma}, s^\rho), \qquad \mathrm{lev}\ (\rho \to \sigma) < k$$

Obviously $\mathrm{NF}_0^{(\mathrm{co})}$ is just the set $\mathrm{NF}^{(\mathrm{co})}$, because cuts at level 0 are not possible.

The functions $r{\uparrow}$ and $r[x]$ carry over from $\mathrm{NF}^{(\mathrm{co})} \subseteq \Lambda^{(\mathrm{co})}$ to $\mathrm{NF}_k^{(\mathrm{co})}$ in the straightforward way.

The NF_k*-size* of terms in NF_k is given recursively by

$$\begin{aligned} |x\vec{r}| &:= 1 + \textstyle\sum |\vec{r}| & \qquad |\mathcal{R}r| &:= 1 + |r| \\ |\lambda r| &:= 1 + |r| & |\beta_s r| &:= 1 + |r| + |s| \\ |\mathcal{C}(r, s)| &:= |r| + |s| \end{aligned}$$

Note that in the variable clause, not all the applications (and thus the length of $\vec{r}$) are counted; instead the addition of 1 serves to mark that one variable application has been passed. Thus $|xy| = 2$. Also remark that the NF_k-size of a cut $\mathcal{C}(r,s)$ is $|r| + |s|$ rather than $1 + |r| + |s|$.

4.4. SN**-derivations.** We define an inductive calculus for deriving terms r of Λ relative to a list of terms $\vec{s}$, using cuts (i.e., applications) up to rank k (written $\vdash_k\ r \mathbin{\square} \vec{s}$, with $\mathbin{\square} \vec{s}$ omitted, if $\vec{s}$ is empty).[8]

For ease of the following arguments we add an explicit notation system for such derivations, using terms of NF_k, so that $t\ \vdash_k\ r \mathbin{\square} \vec{s}$ stands for a derivation of $\vdash_k r \mathbin{\square} \vec{s}$, witnessed by the term t. Furthermore we include a bound on the size of such derivations into the judgement: $t\ \vdash^n_k\ r \mathbin{\square} \vec{s}$ stands for a derivation of size $\leq n$.

$$(\beta)\ \frac{t\ \vdash^n_k\ r[s] \mathbin{\square} \vec{s} \qquad t'\ \vdash^m_k\ s}{\beta_{t'} t\ \vdash^{n+m+1}_k\ \lambda r \mathbin{\square} s, \vec{s}} \qquad\qquad (\lambda)\ \frac{t\ \vdash^n_k r}{\lambda t\ \vdash^{n+1}_k \lambda r}$$

$$(v)\ \frac{\vec{t}\ \vdash^{\vec{n}}_k \vec{r}}{x\vec{t}\ \vdash^{\sum 1,m,\vec{n}}_k\ x \mathbin{\square} \vec{r}} \qquad\qquad (\mathcal{R})\ \frac{t\ \vdash^n_k r \mathbin{\square} s, \vec{s}}{\mathcal{R}t\ \vdash^{n+1}_k\ rs \mathbin{\square} \vec{s}}$$

$$(\mathcal{C})\ \frac{t\ \vdash^n_k\ r \mathbin{\square} \vec{s} \qquad t'\ \vdash^m_k\ s \qquad r\vec{s} : \rho\ \&\ \mathrm{lev}\,\rho \leq k}{\mathcal{C}(t,t')\ \vdash^{n+m}_k\ r \mathbin{\square} \vec{s}, s}$$

Notation. In rule (v) we used the notation $\Sigma\vec{n}$ to add a list of natural numbers.

We abbreviate $t\ \vdash^n_0\ r \mathbin{\square} \vec{s}$ by $t\ \vdash^n\ r \mathbin{\square} \vec{s}$. In accordance with the notation of subsection 3.1 we will omit the size and witness annotations in the judgement whenever reasonable. As further abbreviations define $\mathrm{SN}_{\vec{r},k} := \{r \in \Lambda \mid\ \vdash_k r \mathbin{\square} \vec{r}\}$, $\mathrm{SN}_k := \mathrm{SN}_{\varepsilon,k}$ and $\mathrm{SN} := \mathrm{SN}_0$.

Remarks.

- If $\vec{s} \vdash^n r$ then $n > 0$.
- *Weakening.* The size annotation for the variable rule allows to prove weakening for all derivations:
$$t\ \vdash^n_k\ r \mathbin{\square} \vec{s} \implies t\ \vdash^{n+m}_k\ r \mathbin{\square} \vec{s}.$$
- Note that the premise $\mathrm{lev}\,\rho \leq k$ excludes applications of a cut of rank 0: for r to be applicable it needs to have an arrow type, which has rank > 0. Thus $\mathrm{SN} = \{s \mid \exists t \in \mathrm{NF}.t\ \vdash\ s\}$.
- SN_{k+l} derives all terms of SN_k with exactly the same derivation notations, so that $\mathrm{SN}_k \subseteq \mathrm{SN}_{k+l}$.
- Various inductive definitions for the set of strongly normalizing terms in Λ have been put forward and discussed in the literature (see e.g., [vR96, JM03,Bec01]). The variant presented here differs in its explicit consideration of applications in the $\mathcal{R}$-rule. This enables us to correlate the NF-size of the derivation (notation) with the reduction tree height and the size of the resulting normal form, thus improving estimates for the bounds that accrue.

[8] As suggested by the notation, $\vdash_k$ extends the definition of $\vdash$ in subsection 3.1.

– By comparing the defining rules for @ and SN_0 it becomes clear that a derivation of $\vdash r \mathbin{\square} \vec{s}$ represents a computation tree for $r@\vec{s}$. More precisely, a trivial induction on SN_0 proves the following elaboration of proposition 5:

$$t \ \vdash^n \ r \mathbin{\square} \vec{s} \implies t = r@\vec{s} \ \& \ |t| \leq n.$$

4.5. Closure properties. In this subsection we introduce basic operations on SN_k-derivations that will allow us to derive cuts of rank lower than k by the help of lifting, application and substitution.

Proposition 10. $t \ \vdash^n_k \ r \mathbin{\square} \vec{s} \implies t[m]_l \ \vdash^n_k \ r[m]_l \mathbin{\square} \vec{s}[m]_l \ \& \ t{\uparrow_l} \ \vdash^n_k \ r{\uparrow_l} \mathbin{\square} \vec{s}{\uparrow_l}$.

Definition (Application). The function $a_x : \mathrm{NF}^{\mathrm{co}}_k \to \mathrm{NF}^{\mathrm{co}}_k$ is defined by guarded recursion:

$$\begin{array}{rlrl} a_x(y\vec{t}\,) &:= y\vec{t}x & a_x\lambda t &:= \beta_x.t[x] \\ a_x(\mathcal{R}t) &:= \mathcal{R}.a_x t & a_x(\beta_{t'}t) &:= \beta_{t'}.a_x t \\ a_x\mathcal{C}(t,t') &:= \mathcal{C}(\mathcal{C}(t,t'),x) & & \end{array}$$

Note that $a_x \restriction \mathrm{NF}_k : \mathrm{NF}_k \to \mathrm{NF}_k$.

Lemma 11. $t \ \vdash^n_k \ r \mathbin{\square} \vec{s} \implies a_x t \ \vdash^{n+1}_k \ r \mathbin{\square} \vec{s}, x$.

Proof. The verification of the asserted property of a_x proceeds along the lines of the definition of a_x, using induction on $t : \vec{s} \vdash^n_k \ r$. **Case** (v).

$$\begin{array}{rlll} y\vec{t} \vdash^{\sum 1,m,\vec{n}}_k & y & \mathbin{\square} \vec{s} & \text{from} \\ \vec{t} \vdash^{\sum \vec{n}}_k & & \vec{s}. & \\ x \vdash^1_k & x & & \text{by } (v) \\ y\vec{t}x \vdash^{\sum 2,m,\vec{n}}_k & y & \mathbin{\square} \vec{s}, x & \text{by } (v). \end{array}$$

Case (λ).

$$\begin{array}{rlll} \lambda t \vdash^{n+1}_k & \lambda r & & \text{from} \\ t \vdash^n_k & r. & & \\ t[x] \vdash^n_k & r[x] & & \text{by the prop.} \\ x \vdash^1_k & x & & \text{by } (v) \\ \beta_x t[x] \vdash^{n+2}_k & \lambda r & \mathbin{\square} x & \text{by } (\beta_x). \end{array}$$

Case $(\mathcal{C})$. $t \ \vdash^{n+m}_k \ r \mathbin{\square} \vec{s}, s$ has been concluded by $(\mathcal{C})$. Note that the type of r has level $\leq l$, so in order to be able to apply r to $\vec{s}, s$ and x the variable x has to have type level $< l$. This permits a cut to conclude $\mathcal{C}(t,x) \ \vdash^{n+m+1}_k \ r \mathbin{\square} \vec{s}, s, x$. **Case** $(\mathcal{R})$.

$$\begin{array}{rlll} \mathcal{R}t \vdash^{n+1}_k & rs & \mathbin{\square} \vec{s} & \text{from} \\ t \vdash^n_k & r & \mathbin{\square} s, \vec{s}. & \\ a_x t \vdash^{n+1}_k & r & \mathbin{\square} s, \vec{s}, x & \text{by IH} \\ \mathcal{R}.a_x t \vdash^{n+2}_k & rs & \mathbin{\square} \vec{s}, x & \text{by } (\mathcal{R}). \end{array}$$

Case (β_t). Obvious application of the induction hypothesis. □

Definition (Substitution). The function $b_l : \mathrm{NF}_k^{\mathrm{co}} \times \mathrm{NF}_k^{\mathrm{co}} \longrightarrow \mathrm{NF}_k^{\mathrm{co}}$ is defined by guarded recursion:

$$
\begin{array}{llll}
b_l(\beta_{t_1} t_0, t') := \beta_{b_l(t_1,t')}.b_l(t_0,t') & b_l(\mathcal{R}t, t') & := \mathcal{R}.b_l(t,t') \\
b_l(\lambda t, t') \quad := \lambda b_{l+1}(t, t'\uparrow) & b_l(\mathcal{C}(t_0,t_1),t') & := \mathcal{C}(b_l(t_0,t'), b_l(t_1,t')) \\
b_x(x\vec{t}, t') \quad := \vec{\mathcal{C}}(t', b_l(\vec{t}, t')) & b_l(x\vec{t}, t') & := x[t']_l b_l(\vec{t}, t'), \quad \text{if } l \neq x
\end{array}
$$

To understand the first line of the last clause, note that $x[t']_l$ is again a variable as $x \neq l$. The abbreviation $\vec{\mathcal{C}}(t', \vec{t})$ is to be read $\mathcal{C}(\mathcal{C}(\ldots\mathcal{C}(\mathcal{C}(t',t_1),t_2),\ldots),t_n)$.

Although the definition has been stated for $\mathrm{NF}_k^{\mathrm{co}}$ we remark that $b_l \restriction (\mathrm{NF}_k \times \mathrm{NF}_k) : \mathrm{NF}_k \times \mathrm{NF}_k \longrightarrow \mathrm{NF}_k$.

Lemma 12.
$t \vdash_k^n r \circ \vec{s}$ & $t' \vdash_k^m s^\rho$ & $\mathrm{lev}\,\rho \leq k \implies b_l(t,t') \vdash_k^{n*m} r[s]_l \circ \vec{s}[s]_l$.

Proof. Induction on $t \vdash_k^n r \circ \vec{s}$. **Case** (v). Subcase $l = x$:

$$
\begin{array}{rlll}
x\vec{t} \vdash_k^{\sum 1,n',\vec{n}} & x & \circ\, \vec{r} & \text{from} \\
\vec{t} \vdash_k^{\vec{n}} & \vec{r}. & & \\
b_l(\vec{t},t') \vdash_k^{\vec{n}*m} & \vec{r}[s]_l & & \text{by IH} \\
\vec{\mathcal{C}}(t', b_l(\vec{t},t')) \vdash_k^{m*\sum 1,\vec{n}} & s & \circ\, \vec{r}[s]_l & \text{by } (\mathcal{C}) \\
\vec{\mathcal{C}}(t', b_l(\vec{t},t')) \vdash_k^{m*\sum 1,n',\vec{n}} & s & \circ\, \vec{r}[s]_l & \text{by weakening.}
\end{array}
$$

Subcase $x \neq l$:

$$
\begin{array}{rlll}
x\vec{t} \vdash_k^{\sum 1,n',\vec{n}} & x & \circ\, \vec{r} & \text{from} \\
\vec{t} \vdash_k^{\vec{n}} & \vec{r}. & & \\
b_l(\vec{t},t') \vdash_k^{\vec{n}*m} & \vec{r}[s]_l & & \text{by IH} \\
x[t']_l b_l(\vec{t},t') \vdash_k^{1+n'+\sum \vec{n}*m} & x[t']_l & \circ\, \vec{r}[s]_l & \text{by } (v) \\
x[t']_l b_l(\vec{t},t') \vdash_k^{m*\sum 1,n',\vec{n}} & x[t']_l & \circ\, \vec{r}[s]_l & \text{by weakening.}
\end{array}
$$

Case (β).

$$
\begin{array}{rlll}
\beta_{t_1} t_0 \vdash_k^{n+n'+1} & \lambda r & \circ\, r', \vec{r} & \text{from} \\
t_0 \vdash_k^{n} & r[r'] & \circ\, \vec{r} & \text{and} \\
t_1 \vdash_k^{n'} & r'. & & \\
b_l(t_1,t') \vdash_k^{n'*m} & r'[s]_l & & \text{by IH} \\
b_l(t_0,t') \vdash_k^{n*m} & r[r'][s]_l & \circ\, \vec{r}[s]_l & \text{by IH} \\
b_l(t_0,t') \vdash_k^{n*m} & r[s\uparrow]_{l+1}[r'[s]_l] & \circ\, \vec{r}[s]_l & \text{by lemma 2 (5)} \\
\beta_{b_l(t_1,t')} b_l(t_0,t') \vdash_k^{\sum 1,n*m,n'*m} & \lambda.r[s\uparrow]_{l+1} & \circ\, r'[s]_l, \vec{r}[s]_l & \text{by } (\beta) \\
\beta_{b_l(t_1,t')} b_l(t_0,t') \vdash_k^{(1+n+n')*m} & (\lambda r)[s]_l & \circ\, r'[s]_l, \vec{r}[s]_l & \text{by weakening.}
\end{array}
$$

Case (λ).

$$\begin{array}{rlll} \lambda t \vdash^{n+1}_k & \lambda r & & \text{from} \\ t \vdash^n_k & r. & & \\ t' \vdash^m_k & s & & \text{by assumption} \\ t'\uparrow \vdash^m_k & s\uparrow & & \text{by prop. 10} \\ b_{l+1}(t, t'\uparrow) \vdash^{n*m}_k & r[s\uparrow]_{l+1} & & \text{by IH}_{l+1} \text{ for } s\uparrow \\ \lambda.b_{l+1}(t, t'\uparrow) \vdash^{1+n*m}_k & \lambda.r[s]_{l+1} & & \text{by } (\lambda) \\ \lambda.b_{l+1}(t, t'\uparrow) \vdash^{(1+n)*m}_k & (\lambda r)[s]_l & & \text{by weakening.} \end{array}$$

Case (a).

$$\begin{array}{rlll} \mathcal{R}t \vdash^{n+1}_k & rr' & \circ\, \vec{r} & \text{from} \\ t \vdash^n_k & r & \circ\, r', \vec{r}. & \\ b_l(t, t') \vdash^{n*m}_k & r[s]_l & \circ\, r'[s]_l, \vec{r}[s]_l & \text{by IH} \\ \mathcal{R}b_l(t, t') \vdash^{1+n*m}_k & r[s]_l r'[s]_l & \circ\, \vec{r}[s]_l & \text{by } (\mathcal{R}) \\ \mathcal{R}b_l(t, t') \vdash^{(1+n)*m}_k & (rr')[s]_l & \circ\, \vec{r}[s]_l & \text{by weakening.} \end{array}$$

Case ($\mathcal{C}$).

$$\begin{array}{rlll} \mathcal{C}(t_0, t_1) \vdash^{n+n'}_k & r & \circ\, \vec{r}, r' & \text{from} \\ t_0 \vdash^n_k & r & \circ\, \vec{r} & \text{and} \\ t_1 \vdash^{n'}_k & r'. & & \\ b_l(t_0, t') \vdash^{n*m}_k & r[s]_l & \circ\, \vec{r}[s]_l & \text{by IH} \\ b_l(t_1, t') \vdash^{n'*m}_k & r'[s]_l & \circ & \text{by IH} \\ \mathcal{C}(b_l(t_0, t'), b_l(t_1, t')) \vdash^{(n+n')*m}_k & r[s]_l & \circ\, \vec{r}[s]_l, r'[s]_l & \text{by } (\mathcal{C}). \end{array}$$

□

4.6. Cut elimination. Cuts are eliminated from derivations by successively applying the process of cut reduction.

Definition (Cut reduction). $c : \mathrm{NF}_{k+1} \to \mathrm{NF}_k$ is defined recursively by

$$\begin{array}{llll} c(x\vec{t}) & := x(c\vec{t}) & c\lambda t & := \lambda.ct \\ c\beta_{t'}t & := \beta_{ct'}.ct & c\mathcal{R}t & := \mathcal{R}.ct \\ c\mathcal{C}(t, t') & := b_0(a_0((ct)\uparrow), ct') & & \end{array}$$

c proceeds through the term t and replaces cuts of rank $k+1$ by cuts of rank k, using the functions a and b. Note that c is not a priori well-defined on non-wellfounded terms. Consider e.g., the infinite term $r := \mathcal{C}(0, r)$, which can be typed ρ in the context $0 : \rho \to \rho$ for any ρ; using the clauses above naively we would compute

$$cr = b_0(a_0 1, cr) = b_0(1\,0, cr) = 0cr.$$

Yet c is not continuous, because it needs to explore its argument in quite some depth before yielding a constructor. To see this consider the finite terms

$$r_0 := \lambda 0, \qquad r_{n+1} := \mathcal{C}(r_n, \lambda 0)$$

which are perfectly typable (albeit with increasing types). To compute cr_n, the function c has to explore its argument up to depth n:

$$cr_0 = \lambda 0, \qquad cr_{n+1} = b_0(a_0 cr_n, \lambda 0).$$

Lemma 13. $t \vdash^n_{k+1} r \mathbin{\square} \vec{s} \implies ct \vdash^{2^n-1}_k r \mathbin{\square} \vec{s}$.

Proof. Induction on derivations. We only verify the interesting case of a cut of rank $k+1$:

$$\begin{array}{rlll}
\mathcal{C}(t,t') \vdash^{n+m}_{k+1} & r & \square\, \vec{s}, s & \text{from} \\
t \vdash^n_{k+1} & r & \square\, \vec{s} & \text{and} \\
t' \vdash^m_{k+1} & s. & & \\
ct \vdash^{2^n-1}_k & r & \square\, \vec{s} & \text{by IH} \\
ct' \vdash^{2^m-1}_k & s & & \text{by IH} \\
(ct)\!\uparrow \vdash^{2^n-1}_k & r\!\uparrow & \square\, \vec{s}\!\uparrow & \text{by prop. 10} \\
a_0((ct)\!\uparrow) \vdash^{2^n}_k & r\!\uparrow & \square\, \vec{s}\!\uparrow, 0 & \text{by lemma 11} \\
b_0(a_0((ct)\!\uparrow), ct') \vdash^{2^n*(2^m-1)}_k & (r\!\uparrow)[s] & \square\, (\vec{s}\!\uparrow)[s], 0[s] & \text{by lemma 12} \\
b_0(a_0((ct)\!\uparrow), ct') \vdash^{2^{n+m}-1}_k & r & \square\, \vec{s}, s & \text{by weakening.}
\end{array}$$

□

Definition (Cut elimination). The function $d_k : \mathrm{NF}_k \to \mathrm{NF}$ is defined by recursion on k:

$$d_0 t := t, \qquad d_{k+1} t := d_k ct.$$

Notation. For ordinals α, γ, the expression $\alpha_n \beta$ denotes the α-tower, given by $\alpha_0 \gamma := \gamma$, $\alpha_{n+1}\gamma := \alpha^{\alpha_n \gamma}$.

Theorem 14. $t \vdash^n_k r \mathbin{\square} \vec{s} \implies d_k t \vdash^{2_k n}_0 r \mathbin{\square} \vec{s}$.

Proof. Correctness is verified by induction on k. **Case** 0. Trivial, since $2_0 n = n$. **Case** $k+1$.

$$\begin{array}{rll}
t \vdash^n_{k+1} & r \mathbin{\square} \vec{s}. & \\
ct \vdash^{2^n-1}_k & r \mathbin{\square} \vec{s} & \text{by the previous lemma} \\
ct \vdash^{2^n}_k & r \mathbin{\square} \vec{s} & \text{by weakening} \\
d_k(ct) \vdash^{2_k(2^n)} & r \mathbin{\square} \vec{s} & \text{by IH.}
\end{array}$$

□

4.7. Embedding. In the final step, every term is derived by means of cuts. To this end we need the concept of rank.

Definition (Cut-rank). The rank of a typed term $r \in \vec{\Lambda}$ is given by

$$\mathrm{rk}\ x^\rho := \mathrm{lev}\ \rho,\ \mathrm{rk}\ (rs) := \max(\mathrm{rk}\ r, \mathrm{rk}\ s),\ \mathrm{rk}\ (\lambda^\rho r) := \max(1 + \mathrm{lev}\ \rho, \mathrm{rk}\ r).$$

For non-wellfounded terms, a recursive rank definition is not possible, so that the proof of bounded rank has to be supplied externally. To this end, one has to introduce a coinductive calculus to derive bounds for the rank (written $\mathrm{rk}\ r < k$) as follows:

$$\frac{\mathrm{lev}\ \rho < k}{\mathrm{rk}\ x^\rho < k} \qquad \frac{\mathrm{rk}\ r < k \qquad \mathrm{rk}\ s < k}{\mathrm{rk}\ (rs) < k} \qquad \frac{\mathrm{rk}\ r < k \qquad 1 + \mathrm{lev}\ \rho < k}{\mathrm{rk}\ (\lambda^\rho r) < k}$$

Definition (Embedding). The embedding $[\![r]\!] \in \mathrm{NF}_k^{(\mathrm{co})}$ of terms $r \in \Lambda^{(\mathrm{co})}$ is defined by guarded recursion.

$$[\![x]\!] := x, \quad [\![rs]\!] := \mathcal{RC}([\![r]\!], [\![s]\!]), \quad [\![\lambda r]\!] := \lambda [\![r]\!].$$

Thus for each application one $\mathcal{R}$-rule and one cut is introduced.

The embedding of a term $r \in \Lambda$ is linear in its *size*, defined recursively by

$$\|x\| := 1, \quad \|rs\| := 1 + \|r\| + \|s\|, \quad \|\lambda r\| := 1 + \|r\|.$$

Remark that the notion of size is different from the NF-size. For instance, $\|xyz\| = 5$, while $|xyz| = 3$.

Lemma 15. $\mathrm{rk}\, r \leq k \implies [\![r]\!] \vdash_k^{\|r\|} r.$

Proof. Induction on r. **Case** x. $x = [\![x]\!] \vdash_0^1 x$. **Case** λr. By IH $[\![r]\!] \vdash_k^{|r|} r$, thus $\lambda [\![r]\!] = [\![\lambda r]\!] \vdash_k^{||r||+1} \lambda r$. **Case** rs.

$$\begin{array}{rll}
[\![r, s]\!] \vdash_k^{||r||,||s||} & r, s & \text{by IH} \\
\mathcal{C}([\![r]\!], [\![s]\!]) \vdash_k^{||r||+||s||} & r \quad \square\, s & \text{by } (\mathcal{C}) \\
\mathcal{RC}([\![r]\!], [\![s]\!]) \vdash_k^{1+||r||+||s||} & rs & \text{by } (\mathcal{R}).
\end{array}$$

$\square$

After this embedding we apply cut-elimination (theorem 14) and obtain

Corollary 16. $\mathrm{rk}\, r \leq k \implies d_k [\![r]\!] \vdash^{2_k \|r\|} r.$

Combining all previous results we get

Corollary 17. $\mathrm{rk}\, r \leq k \implies d_k [\![r]\!] = r^\beta \ \& \ |r^\beta| \leq 2_{\mathrm{rk}\, r} \|r\|.$

Remark. The embedding we gave is not the only one possible — another variant for the application clause is $[\![rs]\!] := b_0(a_0([\![r]\!]\uparrow), [\![s]\!])$. However, the size annotations would yield a multiplicative blowup, because the substitution function b_0 leads to the multiplication of sizes. For the following section, where we use height annotations instead, we will invoke the variant to obtain an embedding of r at cut-rank $\mathrm{rk}\, r - 1$.

4.8. Bounds. To complete the analysis, we have to relate the NF-size of r^β to the height of the reduction tree (given recursively by $\#r := \sup\{\#s + 1 \mid r \to s\}$) and the size of the normal form of r. This is the subject of the following

Lemma 18. $\#r\vec{s}^{\,k} + \|\mathrm{nf}\ r\vec{s}\| \leq |r @ \vec{s}| + k.$

Proof. Induction on $|r@\vec{s}|$.

$$\begin{aligned}
\#(\lambda r)s\vec{s} + \|\mathrm{nf}\ (\lambda r)s\vec{s}\| &\leq 1 + \#r[s]\vec{s} + \#s + \|\mathrm{nf}\ r[s]\vec{s}\| \\
&\leq 1 + |r[s]@\vec{s}| + k + \#s && \text{by IH} \\
&\leq 1 + |r[s]@\vec{s}| + k + |s^\beta| && \text{by IH} \\
&= |\beta_{s^\beta}.r[s]@\vec{s}| + k \\
&= |(\lambda r)@(s,\vec{s})| + k. \\
\#\lambda r + \|\mathrm{nf}\ \lambda r\| &= \#r + 1 + \|\mathrm{nf}\ r\| \\
&\leq 1 + |r^\beta| && \text{by IH} \\
&= |\lambda r^\beta| \\
&= |(\lambda r)^\beta|. \\
\#(x\vec{r}) + \|\mathrm{nf}\ (x\vec{r}^{\,k})\| &= \sum \#\vec{r} + 1 + k + \sum \|\vec{r}\| \\
&\leq 1 + k + \sum |\vec{r}^{\,\beta}| && \text{by IH} \\
&= |x\vec{r}^{\,\beta}| + k \\
&= |x@\vec{r}| + k. \\
\#(rs\vec{s}) + \|\mathrm{nf}\ (rs\vec{s})\| &\leq |r@(s,\vec{s})| + k + 1 && \text{by IH} \\
&= |\mathcal{R}.r@(s,\vec{s})| + k \\
&= |rs@\vec{s}| + k.
\end{aligned}$$

□

Corollary 19. $\#r + \|\mathrm{nf}\ r\| \leq 2_{\mathrm{rk}\ r}\|r\|$.

More verbosely, the sum of the height of the reduction tree and the size of the resulting normal form are bounded by the expression $2_{\mathrm{rk}\ r}\|r\|$. This strengthens similar results by Beckmann [BW00], who obtained $2_{\mathrm{rk}\ r}(hr)$ (with h the height of a term) as a bound for the height of the reduction tree alone, without taking the size of the normal form into account.

4.9. Continuous cut-elimination. Although our analysis was concerned with wellfounded terms mainly, all the functions of the cut-elimination process can be applied to non-wellfounded terms as well. The only constituent of the cut-elimination function $d_k[\![r]\!]$ which we cannot prove to be continuous, is the function c. Yet the concatenation $c \circ c \circ \ldots \circ c \circ [\![-]\!]$ is equal to $(-)^\beta$ and thus continuous. This is all the more surprising as only one $\mathcal{R}$ per application is inserted through $[\![-]\!]$ and all following manipulations carefully rearrange these $\mathcal{R}$s by the functions a and b that are called upon by the function c.

So we obtain that cut-elimination is even defined on the set of welltyped non-wellfounded terms with a priori limited cut-rank k.

5 A $\boldsymbol{\lambda}$-calculus with $\boldsymbol{\omega}$-rule

In this section we extend the approach of the previous section to the calculus T_∞, distinguished by the infinitary ω-rule

$$\frac{r \in \mathsf{T}_\infty \qquad \forall n \in \mathbb{N}.s_n \in \mathsf{T}_\infty}{r(s_n)_{n\in\mathbb{N}} \in \mathsf{T}_\infty}$$

5.1. Terms. So terms are given by

$$\mathsf{T}_\infty^{(\mathrm{co})} \ni r, s ::=^{(\mathrm{co})} x \mid rs \mid \lambda r \mid \mathtt{0} \mid \$r \mid r(s_n)_{n\in\mathbb{N}}.$$

Notation. We write $(r_n)_n$ for $(r_n)_{n\in\mathbb{N}}$ and abbreviate $\langle r\rangle := (r_n)_n$. In order to subsume both application and the infinitary elimination syntactically, *eliminations* R, S, T are either terms r or expressions of the form $\langle r\rangle$. This allows the unique display of every wellfounded term in one of the following forms:

$$(\lambda r)\vec{S}, \qquad \mathtt{0}\vec{S}, \qquad (\$r)\vec{S}, \qquad x\vec{S}.$$

5.2. Reductions. Although we will not consider reduction in T_∞ it is helpful to visualize the reductions we have in mind when performing normalization. These include the computational contractions

$$(\lambda r)s \mapsto r[s], \qquad \mathtt{0}\langle s\rangle \mapsto s_0, \qquad (\$r)\langle s\rangle \mapsto r(s_{n+1})_n$$

as well as the following permutative contraction:

$$r\langle s\rangle T \mapsto r(s_n T)_n.$$

Remark. Term closure rules for infinitary calculi are notoriuosly problematic. As for the term closure rules for $r\langle s\rangle$, two variants are conceivable for reduction in $\langle s\rangle$.

$$\frac{\forall n. s_n \to s'_n}{r\langle s\rangle \to r\langle s'\rangle} \qquad \frac{\forall n. s_n \to^* s'_n}{r\langle s\rangle \to r\langle s'\rangle}$$

Our treatment implicitly appeals to the first variant rather than the second. Referring the interested reader to [Maa75] and [Joa01], we note without proof that both notions of reduction are confluent, although this fact will not be needed.

5.3. Continuous normalization. In order to define a continuous normalization function on the infinitary system T_∞, we add the already known repetition constructors βr and $\mathcal{R}r$, as well as an additional repetition constant πr for permutation.

It will turn out that permutations are only required at certain positions in normal forms. More precisely, we abbreviate $\pi^n r := \underbrace{\pi \dots \pi}_{n \text{ times}} r$ and define

$$\mathrm{NF}^{(\mathrm{co})} \ni r, s, t \ ::=^{(\mathrm{co})} \ \pi^n.x\vec{r}\langle r\rangle \mid x\vec{r} \mid \lambda r \mid \mathcal{R}r \mid \beta r \mid \mathtt{0} \mid \$r.$$

The *continuous normalization* function $r@\vec{S}$ normalizes $r \in \mathsf{T}_\infty^{\mathrm{co}}$ relative to a list of eliminations $\vec{S}$, returning an element of $\mathrm{NF}^{\mathrm{co}}$. It is defined by guarded recursion as follows:

$$
\begin{array}{llll}
x \mathbin{@} (\vec{r}, \langle r\rangle, \vec{S}^{\,n}) & := \pi^n.x\vec{r}^{\beta}(r_m@\vec{S})_m, & (\lambda r) \mathbin{@} (s, \vec{S}) & := \beta.r[s]@\vec{S}, \\
x \mathbin{@} \vec{r} & := x\vec{r}^{\beta}, & (\lambda r) \mathbin{@} \varepsilon & := \lambda r^{\beta}, \\
\emptyset \mathbin{@} (\langle r\rangle, \vec{S}) & := \beta.r_0@\vec{S}, & (\$r) \mathbin{@} (\langle r\rangle, \vec{S}) & := \beta.r@((r_{n+1})_n, \vec{S}), \\
\emptyset \mathbin{@} \varepsilon & := \emptyset, & (\$r) \mathbin{@} \varepsilon & := \$r^{\beta}, \\
(rS) \mathbin{@} \vec{S} & := \mathcal{R}.r@(S, \vec{S}), & r^{\beta} & := r@\varepsilon, \\
(\lambda r) \mathbin{@} (\langle r\rangle, \vec{S}) & := \bot, & \emptyset \mathbin{@} (s, \vec{S}) & := \bot, \\
(\$r) \mathbin{@} (s, \vec{S}) & := \bot. & &
\end{array}
$$

using $\bot$ as a symbol for finite failure (can be set to $\emptyset$ or 0 or $\bot := \mathcal{R}\bot$). The last three clauses are required for pathological terms like $(\lambda r)\langle s\rangle$, for which no semantic intuition exists; they will be ruled out by typing altogether.

5.4. Types. The type assignment for the λ-calculus is canonically extended to $\mathsf{T}_{\infty}^{(\mathrm{co})}$ by adding a base type $\mathbb{N}$ and the following rules:

$$
\frac{}{\phi \Vdash \emptyset : \mathbb{N}} \qquad \frac{\phi \Vdash r : \mathbb{N}}{\phi \Vdash \$r : \mathbb{N}} \qquad \frac{\phi \Vdash r : \mathbb{N} \qquad \forall n. \phi \Vdash s_n : \rho}{\phi \Vdash r\langle s\rangle : \rho}
$$

We write $\vec{\mathsf{T}}_{\infty}$ for the typable terms of T_{∞}, given a fixed ϕ.

Obviously all contraction rules preserve typability and types ("subject reduction").

For the rest of this section we assume all mentioned terms to be typable.

5.5. Cuts. We now set out to repeat the cut-elimination argument for $\vec{\Lambda}$ in the extended calculus $\vec{\mathsf{T}}_{\infty}$. This involves the extension of our ambient notion of normal forms by a cut $\mathcal{C}(r, s)$.

Unfortunately, the presence of permutations complicates matters slightly: during cut-elimination it will become necessary to execute an ω-elimination on terms with cuts. To accommodate this, we also introduce a combined constructor $\mathcal{C}_n(r, s, \langle t\rangle)$, corresponding to an ω-elimination (with n following permutations) with side terms $\langle t\rangle$ on a cut-term.

So the semiformal system $\mathrm{NF}_k^{(\mathrm{co})}$ is defined by

$$
\begin{aligned}
\mathrm{NF}_k^{(\mathrm{co})} \ni r, s ::=^{(\mathrm{co})} & \ \pi^m.x\vec{r}\langle r\rangle \mid x\vec{r} \mid \lambda r \mid \mathcal{R}r \mid \beta r \mid \emptyset \mid \$r \\
& \mid \mathcal{C}(r^{\rho\to\sigma}, s) \mid \mathcal{C}_n(r^{\rho\to\mathbb{N}}, s, \langle r\rangle),
\end{aligned}
$$

where the last two forms are subject to the proviso $\mathrm{lev}\,(\rho \to \sigma) < k$ and $\mathrm{lev}\,(\rho \to \mathbb{N}) < k$, respectively.

5.6. Derivations. In section 4 we saw that normal forms with $\mathcal{R}$ and β can be used to denote derivations of strong normalizability for terms in Λ. Analogously, normal forms in NF_k denote derivations of normalizability for T_{∞}-terms. In contrast to the finitary system Λ, however, we can only be interested in weak normalizability, the notion of strong normalization lacking sense in infinitary systems (see the final remark in section 6 for a more elaborate view on this).

Also we will be concerned with the *height* of derivations rather than their size. The height being infinite, it will be measured by ordinals $\alpha, \gamma, \xi, \mu < \varepsilon_0$.

So we define a calculus to derive judgements of the form $t \vdash^\alpha_k r \mathbin{\square} \vec{S}$, where the ordinal α denotes a bound on the height of the derivation, $t \in \mathrm{NF}_k$ and r and $\vec{S}$ are terms and eliminations of T_∞, respectively.

For the following rules assume $\alpha', \alpha'', \alpha_n < \alpha$ and $\vec{\gamma} \leq \alpha$.

$$(\beta)\ \frac{t \vdash^{\alpha'}_k r[s] \mathbin{\square} \vec{S}}{\beta t \vdash^\alpha_k \lambda r \mathbin{\square} s, \vec{S}} \qquad (\lambda)\ \frac{t \vdash^{\alpha'}_k r}{\lambda t \vdash^\alpha_k \lambda r} \qquad (\emptyset)\ \frac{}{\emptyset \vdash^\alpha_k \emptyset}$$

$$(\beta_0)\ \frac{t \vdash^{\alpha'}_k r_0 \mathbin{\square} \vec{S}}{\beta t \vdash^\alpha_k \emptyset \mathbin{\square} \langle r\rangle, \vec{S}} \qquad (\beta_\$)\ \frac{t \vdash^{\alpha'}_k r \mathbin{\square} (s_{n+1})_n, \vec{S}}{\beta t \vdash^\alpha_k (\$r) \mathbin{\square} \langle s\rangle, \vec{S}} \qquad (\$)\ \frac{t \vdash^{\alpha'}_k r}{\$t \vdash^\alpha_k \$r}$$

$$(v)\ \frac{\vec{t} \vdash^{\vec{\gamma}}_k \vec{r}^{\,l}}{x\vec{t} \vdash^{\alpha+l}_k x \mathbin{\square} \vec{r}} \qquad (\mathcal{C})\ \frac{t \vdash^{\alpha'}_k (r \mathbin{\square} \vec{S})^{\rho\to\sigma} \qquad t' \vdash^{\alpha''}_k s}{\mathcal{C}(t, t') \vdash^\alpha_k r \mathbin{\square} \vec{S}, s}$$

$$(\mathcal{R})\ \frac{t \vdash^{\alpha'}_k r \mathbin{\square} S, \vec{S}}{\mathcal{R}t \vdash^\alpha_k rS \mathbin{\square} \vec{S}} \qquad (\pi)\ \frac{\vec{t} \vdash^{\vec{\gamma}}_k \vec{s}^{\,l} \qquad \forall n. t_n \vdash^{\alpha_n}_k r_n \mathbin{\square} \vec{R}^m}{\pi^m.x\vec{t}\langle t\rangle \vdash^{\alpha+l}_k x \mathbin{\square} \vec{s}, \langle r\rangle, \vec{R}}$$

For technical reasons, the remaining rule for $\mathcal{C}_n(t, s, \langle t\rangle)$ requires the additional assumption $\forall m. \alpha_m < \alpha''' < \alpha$:

$$(\mathcal{C}_n)\ \frac{t \vdash^{\alpha'}_k (r \mathbin{\square} \vec{S})^{\rho\to\mathbb{N}} \qquad t' \vdash^{\alpha''}_k s \qquad \forall m. t_m \vdash^{\alpha_m}_k r_m \mathbin{\square} \vec{T}^n}{\mathcal{C}_n(t, s, \langle t\rangle) \vdash^\alpha_k r \mathbin{\square} \vec{S}, s, \langle r\rangle, \vec{T}}$$

The rules for $\mathcal{C}$ and $\mathcal{C}_n$ are subject to the proviso that lev $(\rho \to \sigma) \leq k$ and lev $(\rho \to \mathbb{N}) \leq k$, respectively.

Remarks.

- Scrutiny is required for the rule $\pi^m.x\vec{t}\langle t\rangle \vdash x \mathbin{\square} \vec{s}, \langle r\rangle, \vec{R}$: all the subderivations t_n derive the side terms r_n with the eliminations $\vec{R}$ attached, where the length of $\vec{R}$ is determined by the index m. The same remark applies to the rule for $\mathcal{C}_n$.
- Using induction on $\vdash$ it is straightforward to show that $r\vec{S}$ is weakly normalizing w.r.t. the above mentioned notions of reduction, although this will not be further used. More precisely, one can show
$$t \vdash r \mathbin{\square} \vec{S} \implies r\vec{S} \to^* (t^*),$$
where t^* is the $\mathcal{R}, \beta, \pi$-free form of t (defined analogously to subsection 2.3).
- *Weakening* is admissible: $t \vdash^\alpha_k r \mathbin{\square} \vec{S}$ implies $t \vdash^\gamma_k r \mathbin{\square} \vec{S}$ for $\alpha < \gamma$.

5.7. Lifting and variable substitution. The definition of $t{\uparrow}$ and $t[x]_l$ is canonically extended to the calculus with the new forms: for $\theta \in \{\uparrow, [x]\}$ we set

$$\mathcal{C}_n(t, t', \langle t\rangle)\theta_k := \mathcal{C}_n(t\theta_k, t'\theta_k, (t_n\theta_k)_n) \qquad (\pi^n.y\vec{t}\langle t\rangle)\theta_k := \pi^n.y\theta_k\vec{t}\theta_k(t_n\theta_k)_n$$
$$\emptyset\theta_k := \emptyset \qquad (\$t)\theta_k := \$.t\theta_k$$

Proposition 20. $t \vdash_k^\alpha r \,\square\, \vec{S} \implies t\theta_k \vdash_k^\alpha r\theta_k \,\square\, \vec{S}\theta_k$.

5.8. Admissibility of application. The definition of the function a_x (cf. subsection 4.5) is easily extended to $\mathrm{NF}_k^{(\mathrm{co})}$:

$$\begin{array}{llll} a_x(\pi^n.y\vec{t}\langle t'\rangle) := \pi^{n+1}.y\vec{t}(a_x t'_l)_l, & a_x(y\vec{t}) & := y\vec{t}x, \\ a_x(\lambda t) \quad := \beta.t[x], & a_x(\mathcal{R}t) & := \mathcal{R}.a_x t \\ a_x(\mathcal{C}(t,t')) \quad := \mathcal{C}(\mathcal{C}(t,t'),x)), & a_x(\mathcal{C}_n(t,t',\langle t\rangle)) & := \mathcal{C}_{n+1}(t,t',(a_x t_n)_n) \\ a_x(\beta t) \quad := \beta.a_x t. \end{array}$$

Lemma 21. $t \vdash_k^\alpha r \,\square\, \vec{S} \implies a_x t \vdash_k^{\alpha+1} r \,\square\, \vec{S}, x$.

Proof. Induction. We only treat three cases. **Case** $\pi^n.y\vec{t}\langle t\rangle$.

$$\begin{array}{rlll} \pi^n.y\vec{t}^m\langle t\rangle \vdash_k^{\alpha+m} & y & \square\, \vec{s},\langle r\rangle,\vec{T}^n & \text{from} \\ \vec{t} \vdash_k^{\vec{\alpha}} & \vec{s} & & \text{and} \\ \forall l.t_l \vdash_k^{\alpha''_l} & r_l & \square\, \vec{T}. & \\ \forall l.t_l \vdash_k^{\alpha''_l+1} & r_l & \square\, \vec{T},x & \text{by IH} \\ \pi^{n+1}.y\vec{t}(a_x t_l)_l \vdash_k^{\alpha+m+1} & y & \square\, \vec{s},\langle r\rangle,\vec{T},x & \text{since } \alpha''_l+1 < \alpha+1. \end{array}$$

Case $\mathcal{C}_n(t,t',\langle t\rangle)$.

$$\begin{array}{rlll} \mathcal{C}_n(t,t',\langle t\rangle) \vdash_k^{\alpha} & r & \square\, \vec{S},s,\langle r\rangle,\vec{T}^n & \text{from} \\ t \vdash_k^{\alpha'} & r & \square\, \vec{S} & \text{and} \\ t' \vdash_k^{\alpha''} & s & & \text{and} \\ \forall l.t_l \vdash_k^{\alpha_l} & r_l & \square\, \vec{T} & \text{with } \alpha_l < \alpha''' < \alpha. \\ \forall l.a_x t_l \vdash_k^{\alpha_l+1} & r_l & \square\, \vec{T},x & \text{by IH} \\ \mathcal{C}_n(t,t',(a_x t_l)_l) \vdash_k^{\alpha+1} & r & \square\, \vec{S},s,\langle r\rangle,\vec{T},x & \text{by } (\mathcal{C}_n). \end{array}$$

Case $y\vec{t}^m$.

$$\begin{array}{rlll} y\vec{t}^m \vdash_k^{\alpha+m} & y & \square\, \vec{s} & \text{from} \\ \vec{t} \vdash_k^{\vec{\gamma}} & \vec{s}. & & \\ x \vdash_k^0 & x & & \text{by } (v). \\ y\vec{t}x \vdash_k^{\alpha+m+1} & y & \square\, \vec{s},x & \text{by } (v). \end{array}$$

□

5.9. Admissibility of the ω-rule. In this subsection we reconstruct the infinitary elimination on derivations. It is necessary to include a list of n permutations into the definition of the witnessing functional $p_n(r,\langle s\rangle)$, which takes an argument of type $\mathbb{N}$ and an elimination.

$$\begin{array}{ll} p_n(x\vec{r},\langle s\rangle) & := \pi^n.x\vec{r}\langle s\rangle, \\ p_n(\emptyset,\langle s\rangle) & := \beta.s_0, \\ p_n(\pi^m.x\vec{r}\langle r\rangle,\langle s\rangle) & := \pi^{n+m+1}.x\vec{r}(p_n(r_k,\langle s\rangle))_k, \\ p_n(\$r,\langle s\rangle) & := \beta.p_n(r,(s_{k+1})_k), \\ p_n(\mathcal{R}r,\langle s\rangle) & := \mathcal{R}.p_n(r,\langle s\rangle), \\ p_n(\beta r,\langle s\rangle) & := \beta.p_n(r,\langle s\rangle), \\ p_n(\mathcal{C}(r_0,r_1),\langle s\rangle) & := \mathcal{C}_0(r_0,r_1,\langle s\rangle), \\ p_n(\mathcal{C}_m(\hat{t},\tilde{t},\langle t\rangle),\langle t'\rangle) & := \mathcal{C}_{n+m+1}(\hat{t},\tilde{t},(p_n(t_l,\langle t'\rangle))_l). \end{array}$$

Lemma 22.
$t \vdash_k^\alpha r \,\square\, \vec{S} \;\&\; \forall n(t'_n \vdash_k^{\gamma_n} s_n \,\square\, \vec{T}^m \;\&\; \gamma_n < \gamma) \implies p_m(t, \langle t' \rangle) \vdash_k^{\gamma+\alpha} r \,\square\, \vec{S}, \langle s \rangle, \vec{T}$.

Proof. Induction on $t \vdash_k^\alpha r \,\square\, \vec{S}$. We only illustrate some cases. **Case** $(\mathcal{C}_m)$.

$$\begin{array}{rlll}
\mathcal{C}_{m'}(\hat{t}, \tilde{t}, \langle t \rangle) \vdash_k^\alpha & r & \square\, \vec{S}, s, \langle s' \rangle, \vec{S}'^{\,m'} & \text{from} \\
\forall l. t_l \vdash_k^{\alpha_l} & s'_l & \square\, \vec{S}' & \text{with } \alpha_l < \alpha''' < \alpha \text{ and} \\
\hat{t} \vdash_k^{\alpha'} & r & \square\, \vec{S} & \text{and} \\
\tilde{t} \vdash_k^{\alpha''} & s. & & \\
p_m(t_l, \langle t' \rangle) \vdash_k^{\gamma+\alpha_l} & s'_l & \square\, \vec{S}', \langle s \rangle, \vec{T} & \text{by IH} \\
\mathcal{C}_{m+m'+1}(\hat{t}, \tilde{t}, (p_m(t_l, \langle t' \rangle))_l) \vdash_k^{\gamma+\alpha} & r & \square\, \vec{S}, s, \vec{S}', \langle s \rangle, \vec{T} & \text{by } (\mathcal{C}) \\
 & & & \text{since } \gamma{+}\alpha_l < \gamma{+}\alpha''' < \gamma{+}\alpha.
\end{array}$$

Case $\pi^m.x\vec{t}^{\,l}\langle t \rangle$.

$$\begin{array}{rlll}
\pi^m.x\vec{t}\langle t \rangle \vdash_k^{\alpha+l} & x & \square\, \vec{S}, \langle r \rangle, \vec{R}^n & \text{from} \\
\vec{t} \vdash_k^{\vec{\alpha}} & \vec{S} & & \text{with } \vec{\alpha} \leq \alpha \text{ and} \\
\forall l. t_l \vdash_k^{\alpha_l} & r_l & \square\, \vec{R} & \text{with } \alpha_l < \alpha. \\
\forall l. p_n(t_l, \langle t' \rangle) \vdash_k^{\gamma+\alpha_l} & r_l & \square\, \vec{R}, \langle s \rangle, \vec{T} & \text{by IH} \\
\pi^{m+n+1}.x\vec{t}(p_n(t_l, \langle t' \rangle)) \vdash_k^{\gamma+\alpha+l} & r & \vec{S}, \langle r \rangle, \vec{R}, \langle s \rangle, \vec{T}. &
\end{array}$$

Case $\mathcal{C}(t, t')$.

$$\begin{array}{rlll}
\mathcal{C}(t, t') \vdash_k^\alpha & r & \square\, \vec{S}, s & \text{from} \\
t \vdash_k^{\alpha'} & r & \square\, \vec{R} & \text{and} \\
t' \vdash_k^{\alpha''} & s. & & \\
\mathcal{C}_0(t, t', \langle t' \rangle) \vdash_k^{\gamma+\alpha} & r & \square\, \vec{S}, s, \langle s \rangle, \vec{T} & \text{by } (\mathcal{C}_n), \text{ using } \gamma_l < \gamma < \gamma{+}\alpha, \\
 & & & \text{since } \alpha > 0 \text{ as } 0 \leq \alpha' < \alpha.
\end{array}$$

Case $\emptyset$.

$$\begin{array}{rlll}
\emptyset \vdash_k^\alpha & \emptyset. & & \\
t'_0 \vdash_k^{\gamma_0} & s_0 & \square\, \vec{T} & \text{by assumption.} \\
\beta t'_0 \vdash_k^{\gamma} & \emptyset\langle s \rangle & \square\, \vec{T} & \text{by } (\beta_0), \text{ using } \gamma_0 < \gamma.
\end{array}$$

Case $x\vec{t}$.

$$\begin{array}{rlll}
x\vec{t}^{\,l} \vdash_k^{\alpha+l} & x & \square\, \vec{r} & \text{from} \\
\vec{t} \vdash_k^{\vec{\alpha}} & \vec{r}. & & \\
t'_n \vdash_k^{\gamma_n} & s_n & \square\, \vec{T} & \text{by assumption.} \\
\pi^m.x\vec{t}\langle t' \rangle \vdash_k^{\gamma+\alpha+l} & x & \square\, \vec{r}, \langle s \rangle, \vec{T} & \text{by } (\pi), \text{ using } \vec{\alpha} \leq \alpha \leq \gamma{+}\alpha.
\end{array}$$

$\square$

5.10. Admissibility of substitution. Using a_x and p_n we can now establish substitution on $\mathrm{NF}_k^{(\mathrm{co})}$ in the same way as in subsection 4.5.

$$
\begin{array}{ll}
b_x(\beta t, t') & := \beta.b_x(t_0, t'),\\
b_x(\mathcal{R}t, t') & := \mathcal{R}.b_x(t, t')\\
b_x(\lambda t, t') & := \lambda b_{l+1}(t, t'\uparrow)\\
b_x(\mathcal{C}(t_0, t_1), t') & := \mathcal{C}(b_x(t_0, t'), b_x(t_1, t'))\\
b_x(\mathcal{C}_n(t_0, t_1, \langle t\rangle), t') & := \mathcal{C}_n(b_x(t_0, t'), b_x(t_1, t'), (b_x(t_m, t'))_m),\\
b_x(\pi^n.x\langle t\rangle, t') & := p_n(t', (b_x(t_l, t'))_l),\\
b_x(\pi^n.x\vec{t}t\langle t''\rangle, t') & := \mathcal{C}_n(\vec{\mathcal{C}}(t', b_x(\vec{t}, t')), b_x(t, t'), (b_x(t''_m, t'))_m),\\
b_x(\pi^n.l\vec{t}\langle t''\rangle, t') & := \pi^n.l[t']_x b_x(\vec{t}, t')(b_x(r_m, t'))_m, \qquad \text{if } l \neq x,\\
b_x(x\vec{t}, t') & := \vec{\mathcal{C}}(t', b_x(\vec{t}, t')),\\
b_x(l\vec{t}, t') & := l[t']_x b_x(\vec{t}, t'), \qquad \text{if } l \neq x.
\end{array}
$$

Lemma 23. $\tilde{\alpha} < \alpha \ \&\ \tilde{\gamma} < \gamma \implies$
$t \vdash_k^{\tilde{\alpha}} r \circ \vec{S} \ \&\ t' \vdash_k^{\tilde{\gamma}} s^{\rho} \ \&\ \mathrm{lev}\, \rho \leq k \implies \exists \xi < \gamma \cdot \alpha . b_x(t, t') \vdash_k^{\xi} r[s]_x \circ \vec{S}[s]_x$.

Proof. Induction on $t \vdash_k^{\tilde{\alpha}} r \circ \vec{S}$. **Case** $\pi^n.x\langle t\rangle$.

$$
\begin{array}{rlll}
\pi^n.x\langle t\rangle \vdash_k^{\tilde{\alpha}} & r & \circ\, \vec{S}, \langle r\rangle, \vec{T}^n & \text{from}\\
\forall l. t_l \vdash_k^{\alpha_l} & r_l & \circ\, \vec{T}. & \\
\forall l. b_x(t_l, t') \vdash_k^{\xi_l} & r_l[s]_x & \circ\, \vec{T}[s]_x & \text{by IH with } \xi_l < \gamma \cdot \alpha_l,\\
p_n(t', (b_x(t_l, t'))_l) \vdash_k^{\gamma\cdot\tilde{\alpha}+\tilde{\gamma}} & s & \circ\, (r_l[s]_l)_l, \vec{T}[s]_x & \text{by lemma 22.}
\end{array}
$$

To complete the argument we compute

$$\gamma \cdot \tilde{\alpha} + \tilde{\gamma} < \gamma \cdot \tilde{\alpha} + \gamma = \gamma \cdot (\tilde{\alpha}+1) \leq \gamma \cdot \alpha.$$

Case $\pi^n.x\vec{t}^{\,m}t\langle t''\rangle$. Let $\tilde{\alpha}+m+1 < \alpha$.

$$
\begin{array}{rlll}
\pi^n.x\vec{t}t\langle t''\rangle \vdash_k^{\tilde{\alpha}+m+1} & r & \circ\, \vec{s}, s', \vec{S}, \langle r\rangle, \vec{T}^n & \text{from}\\
\forall l. t''_l \vdash_k^{\alpha'_l} & r_l & \circ\, \vec{T} & \text{with } \alpha'_l < \tilde{\alpha} \text{ and}\\
\vec{t}, t \vdash_k^{\vec{\alpha}, \alpha_0} & \vec{s}, s' & & \text{with } \alpha_0, \vec{\alpha} \leq \tilde{\alpha}.\\
\forall l. b_x(t''_l, t') \vdash_k^{\xi'_l} & r_l[s]_x & \circ\, \vec{T}[s]_x & \text{by IH with } \xi'_l < \gamma \cdot \tilde{\alpha}\\
b_x(\vec{t}, t') \vdash_k^{\vec{\xi}} & \vec{s}[s]_x & & \text{by IH with } \vec{\xi} < \gamma \cdot (\tilde{\alpha}+1)\\
b_x(t, t') \vdash_k^{\xi'} & s'[s]_x & & \text{by IH with } \xi' < \gamma \cdot (\tilde{\alpha}+1).
\end{array}
$$

Subcase $m = 0$:

$$\mathcal{C}_n(t', b_x(t, t'), (b_x(t''_m, t'))_m) \vdash_k^{\gamma\cdot(\tilde{\alpha}+1)} r[s]_x \circ \vec{s}[s]_x, s'[s]_x, \vec{S}[s]_x, \langle r\rangle[s]_x, \vec{T}[s]_x.$$

Subcase $m > 0$: then $\vec{t} = t'', \vec{t}^{\,m-1}$ and

$$\mathcal{C}(t', b_x(t'', t')) \vdash_k^{\gamma\cdot(\tilde{\alpha}+1)} s \circ s_1[s]_x$$

and therefore by repeated applications of $(\mathcal{C})$

$$\hat{t} := \vec{\mathcal{C}}(\mathcal{C}(t', b_x(t'', t')), b_x(\vec{t}', t')) \vdash_k^{\gamma\cdot(\tilde{\alpha}+1)+(m-1)} s \circ \vec{s}[s]_x.$$

Thus by $(\mathcal{C}_n)$

$$\mathcal{C}_n(\hat{t}, b_x(t,t'), (b_x(t''_m,t'))_m) \vdash_k^{\gamma\cdot(\tilde{\alpha}+1)+m} r[s]_x \,\square\, \vec{s}[s]_x, s'[s]_x, \vec{S}[s]_x, \langle r\rangle[s]_x, \vec{T}[s]_x.$$

The claim follows from

$$\gamma\cdot(\tilde{\alpha}+1)+m \le \gamma\cdot(\tilde{\alpha}+m+1) < \gamma\cdot\alpha.$$

Note that in both subcases the derivations $b_x(t''_l, t')$ are uniformly bounded by $\gamma\cdot\tilde{\alpha}$ and hence the application of $(\mathcal{C}_n)$ is justified.

All the other cases are similar to those in the proof of lemma 12 and therefore easier. $\square$

5.11. Cut-reduction. Cut-reduction works just as it did for the λ-calculus, except for rule $\mathcal{C}_n$, where we invoke the admissibility of the ω-rule:

$$\begin{array}{llll}
c(x\vec{t}) & := xc\vec{t}, & c(\lambda r) & := \lambda.cr, \\
c(\beta r) & := \beta.cr, & c(\mathcal{R}r) & := \mathcal{R}.cr, \\
c(\mathcal{C}(t,t')) & := b_0(a_0((ct)\uparrow), ct'), & c(\mathcal{C}_n(t_0,t_1,\langle t\rangle)) & := p_n(b_0(a_0((ct)\uparrow),(ct_n)_n), \\
c(\pi^m.x\vec{r}\langle s\rangle) & := \pi^m.x(c\vec{r})(cs_n)_n. & &
\end{array}$$

Lemma 24. *$t \vdash_{k+1}^{\alpha} r \,\square\, \vec{S} \implies ct \vdash_k^{3^{(3^\alpha)}} r \,\square\, \vec{S}$.*

Proof. We only treat the interesting new case $\mathcal{C}_n(t_0,t_1,\langle t'\rangle)$.

$$\begin{array}{rlll}
\mathcal{C}_n(t_0,t_1,\langle t'\rangle) \vdash_{k+1}^{\alpha} & r & \square\, \vec{S}, s, \langle s\rangle, \vec{T}^n & \text{from} \\
t_0 \vdash_{k+1}^{\alpha'} & r & \square\, \vec{S} & \text{and} \\
t_1 \vdash_{k+1}^{\alpha''} & s & & \text{and} \\
\forall l. t'_l \vdash_{k+1}^{\alpha'_l} & s_l & \square\, \vec{T} & \text{with } \alpha'_l < \alpha''' < \alpha. \\
ct_0 \vdash_k^{3^{(3^{\alpha'})}} & r & \square\, \vec{S} & \text{by IH} \\
ct_1 \vdash_k^{3^{(3^{\alpha''})}} & s & & \text{by IH} \\
\forall l. ct'_l \vdash_k^{3^{(3^{\alpha'_l})}} & s_l & \vec{T} & \text{by IH.} \\
a_0((ct_0)\uparrow) \vdash_k^{3^{(3^{\alpha'})}+1} & r\uparrow & \square\, \vec{S}\uparrow, 0 & \text{by prop. 20} \\
t' := b_0(a_0((ct_0)\uparrow), ct_1) \vdash_k^{\xi} & r & \square\, \vec{S}, s & \text{by lemma 23.}
\end{array}$$

We compute

$$\begin{aligned}
\xi &< (3^{(3^{\alpha''})}+1)\cdot(3^{(3^{\alpha'})}+2) && \text{by the lemma} \\
&\le 3^{3^{\alpha''}+1}\cdot 3^{3^{\alpha'}+1} \\
&= 3^{3^{\alpha''}+1+3^{\alpha'}+1} \\
&\le 3^{3^{\alpha''}+3^{\alpha'}+2}.
\end{aligned}$$

Thus lemma 22 yields $p_n(t', (ct'_n)_n) \vdash_k r \,\square\, \vec{S}, s, \langle s \rangle, \vec{T}$ with height

$$\begin{aligned} 3^{3^{\alpha'''}} + \xi &< 3^{3^{\tilde\alpha}+3^{\tilde\alpha}+2} \cdot 2 \\ &\le 3^{3^{\tilde\alpha}+3^{\tilde\alpha}+3} \\ &\le 3^{3^{\tilde\alpha}+3^{\tilde\alpha}+3^{\tilde\alpha}} \\ &= 3^{(3^{\tilde\alpha+1})} \\ &\le 3^{(3^{\alpha})} \end{aligned}$$

where $\tilde\alpha := \max\{\alpha', \alpha'', \alpha'''\} > 0$, because $\alpha''' > \alpha''_l$. □

5.12. Cut-elimination. Set $d_0 t := t, \quad d_{k+1} t := c d_k t$.

Lemma 25. $t \vdash^{\alpha}_{k} r \,\square\, \vec{S} \implies d_k t \vdash^{3_{2k}(\alpha)} r \,\square\, \vec{S}$.

Remark. Comparing this result with the corresponding statements in [BW00] and [Wei97], the ordinal $3_{2k}(\alpha)$ seems unnecessarily big (Weiermann obtains $2_k(\alpha)$ as a bound). This disparity stems from the fact that in our treatment open rather than closed terms are considered and consequently permutative contractions are required which lead to a multiplicative blowup in the size of derivations (cf. lemma 5.9).

The advantage of the approach in this article lies in the particularly strong notion of normal form that holds for open terms of arbitrary type.

For the instance of number-theoretic functions of the form $\lambda t : \mathbb{N} \to \mathbb{N}$ the normal form of t (with all repetition constants removed) is given by the grammar

$$t ::= \emptyset \mid \$t \mid 0 \mid 0\langle t \rangle.$$

We can thus define the value $vt \in \mathbb{N}^{\mathbb{N}}$ of λt by recursion on t as follows

$$\begin{aligned} v\emptyset &:= \boldsymbol{\lambda} x.0, & v(\$t) &:= \boldsymbol{\lambda} x.1 + (vt)x, \\ v0 &:= \boldsymbol{\lambda} x.x, & v(0\langle t \rangle) &:= \boldsymbol{\lambda} x.(vt_x)x. \end{aligned}$$

6 Gödel's T

In this section we embed the iterative version of Gödel's T into the calculus T_∞ in order to derive normalization for it.

6.1. System T. Apart from the basic constructors of the λ-calculus, system T contains $\emptyset$ and the successor constructor $\$r$, as well as $r(s,t)$, standing for r-fold iteration of the function s with starting term t.

$$\mathsf{T}^{(\mathrm{co})} \ni r, s ::=^{(\mathrm{co})} x \mid rs \mid \lambda r \mid \emptyset \mid \$r \mid r(s,t).$$

The *numeral* $\underline{n}$ is defined by recursion on n: $\underline{0} := \emptyset$, $\underline{n{+}1} := \$\underline{n}$. We also set $\underline{\mathbb{N}} := \{\underline{n} \mid n \in \mathbb{N}\}$. Note that recursion $r(s,t)$ is allowed for all terms r rather than just numeralls.

6.2. Reduction. Apart from the β-contraction of the λ-calculus, T contains two further computational contraction rules to model iteration

$$\mathtt{0}(s,t) \mapsto t, \qquad (\$r)(s,t) \mapsto r(s,st).$$

The reduction relation $\to$ is obtained from the contraction rules by means of the term closure.

Remarks.

- The reduction rule for the successor differs slightly from the more usual form $(\$r)(s,t) \to s(r(s,t))$. If iteration is restricted to numerals $\underline{n}(s,t)$ rather than arbitrary terms (as most of the expositions on Gödel's T do anyway and is sufficient for proof theoretic analysis), the two variants coincide, so that they are equi-consistent.
- Using a straightforward encoding of pairs, recursion becomes admissible in T [Sch77].

6.3. Types. The extension of the typing rules to T is immediate:

$$\frac{}{\phi \Vdash \mathtt{0} : \mathbb{N}} \qquad \frac{\phi \Vdash r \ : \mathbb{N}}{\phi \Vdash \$r \ : \mathbb{N}} \qquad \frac{\phi \Vdash r \ : \ \mathbb{N} \qquad \phi \Vdash s \ : \ \rho \to \rho \qquad \phi \Vdash t \ : \ \rho}{\phi \Vdash r(s,t) \ : \ \rho}$$

We use $\vec{\mathsf{T}}$ for the set of typable terms in T. For the rest of this section we assume all mentioned terms to be typable.

6.4. $\mathsf{T} \subset \mathsf{T}_\infty$. Using the abbreviation $r(s,t) := r(s^n t)_n$ inside $\mathsf{T}_\infty^{(\mathrm{co})}$ (with $s^n t := s(\ldots(st))$, n times) every term in $\mathsf{T}^{(\mathrm{co})}$ embeds into $\mathsf{T}_\infty^{(\mathrm{co})}$. This embedding actually preserves contractions; for instance

$$(\$r)(s,t) = (\$r)(s^n t)_n \mapsto r(s^{n+1}t)_n = r(s^n(st))_n = r(s,st).$$

Lemma 26. *$t \vdash r \circ \vec{S}$ & $(r\vec{S})^{\mathbb{N}} \in \vec{T}$ closed $\implies$ $r\vec{S} \to^* t^*$.*

Proof. Induction. We only illustrate the case $(\beta_\$)$ where $\beta t \vdash (\$r) \circ \langle s\rangle, \vec{S}$ has been derived. For $(\$r)\langle s\rangle\vec{S}$ to be in T, $\langle s\rangle$ has to have the form (s,t). Thus $r(s_{n+1})_n\vec{S} = r(s,st)\vec{S}$ and we get $(\$r)(s,t)\vec{S} \to r(s,st)\vec{S} \to^* t^* = (\beta t)^*$ by induction hypothesis. $\square$

Remark. Using the second variant of term closure for $r\langle s\rangle$ given in subsection 5.2, also full reductions in T can be simulated in T_∞.

6.5. Embedding. The term embeddings are NF_k-derivable, using

$$\begin{array}{llll} [x] & := x, & [rs] & := \mathcal{R}.b_0(a_0([r]\uparrow),[s]), \\ [\lambda r] & := \lambda[r], & [\mathtt{0}] & := \mathtt{0}, \\ [\$r] & := \$[r], & [r(s,t)] & := \mathcal{R}.p_0([r],([s^n t])_n) \end{array}$$

Proposition 27. *If μ is a limit ordinal, $\gamma < \mu$, $\xi_0 < \alpha$, $[\![r]\!] \vdash^{\gamma} r$ and $[\![s]\!] \vdash^{\xi_0} s$ then there exists a $\xi_n < \alpha \cdot \mu^n$ such that $[\![r^n s]\!] \vdash^{\xi_n} r^n s$.*

Proof. Induction on n. The case $n = 0$ is trivial. For $n+1$ we note that $[\![r^{n+1}s]\!] = [\![r(r^n s)]\!] = \mathcal{R}.b_0(a([\![r]\!]\uparrow), [\![r^n s]\!])$.

$$\begin{array}{rll}
[\![r]\!] \vdash^{\gamma} & r & \text{by assumption} \\
a_0([\![r]\!]\uparrow) \vdash^{\gamma+1} & r \mathbin{\scriptstyle\square} x & \text{by lemma 5.8} \\
[\![r^n s]\!] \vdash^{\xi_n} & r^n s & \text{by IH with } \xi_n < \alpha \cdot \mu^n \\
b_0(a_0([\![r]\!]\uparrow), [\![r^n s]\!]) \vdash^{\xi} & r \mathbin{\scriptstyle\square} (r^n s) & \text{by lemma 5.10 with } \xi < \xi_n \cdot (\gamma{+}2) \\
\mathcal{R}.b_0(a_0([\![r]\!]\uparrow), [\![r^n s]\!]) \vdash^{\xi_n \cdot (\gamma+2)} & r^{n+1}s & \text{by } (\mathcal{R}).
\end{array}$$

The claim follows from

$$\xi_n \cdot (\gamma + 2) \leq \alpha \cdot \mu^n \cdot (\gamma + 2) < \alpha \cdot \mu^n \cdot \mu,$$

where in the last inequality we used that $\gamma + 2 < \mu$, because $\gamma < \mu$ and μ is a limit ordinal. □

Lemma 28. $\operatorname{rk} r < k \implies \exists l. [\![r]\!] \vdash_k^{\omega^{(\omega^l)}} r.$

Proof. Induction on r. We only consider the case $r(s,t)$, where by induction hypothesis there exists an l with $[\![r]\!], [\![s]\!], [\![t]\!] \vdash_k^{\omega^{(\omega^l)}} r, s, t$. By the proposition

$$[\![s^n t]\!] \vdash^{\omega^{(\omega^l)} \cdot (\omega^{(\omega^l)})^n} s^n t.$$

Since $(\omega^{(\omega^l)})^n < (\omega^{(\omega^l)})^\omega = \omega^{(\omega^{l+1})}$ we can compute

$$\omega^{(\omega^l)} \cdot (\omega^{(\omega^l)})^n < \omega^{\omega^l + \omega^{l+1}} = \omega^{(\omega^{l+1})} < \omega^{(\omega^{l+1})} + 1,$$

so lemma 5.9 yields

$$p_0([\![r]\!], ([\![s^n t]\!])_n) \vdash_k^{\omega^{(\omega^{l+1})} + 1 + \omega^{(\omega^l)}} r \mathbin{\scriptstyle\square} (s,t)$$

and one application of $(\mathcal{R})$ shows

$$\mathcal{R}.p_0([\![r]\!], ([\![s^n t]\!])_n) \vdash_k^{\omega^{(\omega^{l+1})} + 1 + \omega^{(\omega^l)} + 1} r \mathbin{\scriptstyle\square} (s,t).$$

The claim follows from

$$\omega^{(\omega^{l+1})} {+} 1 {+} \omega^{(\omega^l)} {+} 1 < \omega^{(\omega^{l+2})}.$$

□

The reader is invited to compare this with the respective embedding of Λ which led to one cut rank higher.

Corollary 29. $\operatorname{rk} r < k \implies [\![r]\!] \vdash_k^{\omega^{(\omega^\omega)}} r.$

Using the abbreviation $\omega_n := \omega_n(1)$ we obtain

Theorem 30. $\operatorname{rk} r < k \implies d_k[\![r]\!] \vdash^{\omega_{2k+3}} r.$

Combining this with lemma 26 we obtain

Corollary 31. $r^{\mathbb{N}} \in \vec{T}$ *closed* & $\operatorname{rk} r < k \implies r \to^* (d_k[\![r]\!])^* \in \underline{\mathbb{N}}$.

Note that we have only used transfinite induction up to ε_0; all the remaining argument of this normalization proof is formalizable within primitive recursive arithmetic, so the result is obtained in a system of minimal strength.

Remark. In the interest of keeping the presentation technically simple and short we only showed weak normalization for T. Using a similar twist as in subsection 4.1, i.e., adding an index S to the β-constructor and modifying the rules (β), (β_0) and $(\beta_\$)$ as follows

$$(\beta)\ \frac{t \vdash_k^{\alpha'} r[s] \sqcap \vec{S} \qquad t' \vdash_k^{\alpha'} s}{\beta_{t'} t \vdash_k^{\alpha} \lambda r \sqcap s, \vec{S}} \qquad (\beta_0)\ \frac{\forall n. t_n \vdash_k^{\alpha'} r_n \sqcap \vec{S}}{\beta_{\langle t \rangle} t \vdash_k^{\alpha} \mathbb{0} \sqcap \langle r \rangle, \vec{S}}$$

$$(\beta_\$)\ \frac{t \vdash_k^{\alpha'} r \sqcap (s_{n+1})_n, \vec{S} \qquad t' \vdash_k^{\alpha'} s_0 \sqcap \vec{S}}{\beta t \vdash_k^{\alpha} (\$r) \sqcap \langle s \rangle, \vec{S}}$$

one can repeat the argument of this section without major change. Although the concept of strong normalization does not make sense in the infinitary calculus T_∞, its inductive characterization represented by the definition of $t \vdash r$ is still of value to derive strong normalization for T. In this respect, $t \vdash r$ is the correct generalization of the concept of strong normalization to infinitary calculi.

7 Conclusions

The simplified version of continuous normalization used in this article permits a perspicuous analysis of its operational behaviour. Furthermore, by means of the surprising connection between $\mathcal{R}$-annotated normal forms and derivations of normalizability it became possible to improve previously known bounds of the reduction tree height.

There are, however, many interesting properties of continuous normal forms that could not be explored in sufficient detail in this article. Rather than a crude "no", continuous normalization provides precise information about the reduction behaviour of diverging terms. It is therefore possible to capture interesting term classes — such as the class of fixed point operators — through a characterization of their continuous normal forms. For instance, a classification of fixed point operators according to their efficiency or a coinductive analogue of Sørensen's ω-theorem [vRSSX99] seems feasible.

References

[AJ02] Klaus Aehlig and Felix Joachimski. On continuous normalization. In J. Bradfield, editor, *Proc. 11^{th} CSL 2002 (Edinburgh)*, volume 2471 of *Lecture Notes in Computer Science*, pages 59–73. Springer, 2002.

[Bar77] Henk Barendregt. The type free lambda calculus. In Jon Barwise, editor, *Handbook of Mathematical Logik*, chapter D.7, pages 1091–1132. North-Holland Publishing Company, 1977.

[Bec01] Arnold Beckmann. Exact bounds for lengths of reductions in typed λ-calculus. *Journal of Symbolic Logic*, 66(3):1277–1285, 2001.

[Bru72] N.G. de Bruijn. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church–Rosser theorem. *Indagationes Mathematicae*, 34:381–392, 1972.

[Buc91] Wilfried Buchholz. Notation systems for infinitary derivations. *Archive for Mathematical Logic*, 30:277–296, 1991.

[BW00] Arnold Beckmann and Andreas Weiermann. Analyzing Gödel's T via expanded head reduction trees. *Mathematical Logics Quarterly*, 46:517–536, 2000.

[Coq94] Thierry Coquand. Infinite objects in type theory. In H. Barendregt and T. Nipkow, editors, *Proc. 1^{st} Types 1993 (Nijmengen)*, volume 806 of *Lecture Notes in Computer Science*, pages 62–78. Springer, 1994.

[FPT99] Marcelo Fiore, Gordon Plotkin, and Daniele Turi. Abstract syntax and variable binding (extended abstract). In *Proc. 14^{th} LICS 1999 (Trento)*, pages 193–202. IEEE Computer Science Press, 1999.

[Göd58] Kurt Gödel. Über eine bisher noch nicht benützte Erweiterung des finiten Standpunkts. *Dialectica*, 12:280–287, 1958.

[Gim95] Eduardo Gimenez. Codifying guarded definitions with recursive schemes. In J. Smith, B. Nordström, and P. Dybjer, editors, *Proc. Types 1994 (Bastad)*, volume 996 of *Lecture Notes in Computer Science*, pages 35–59. Springer, 1995.

[JM00] Felix Joachimski and Ralph Matthes. Standardization and confluence for a lambda calculus with generalized applications. In Leo Bachmair, editor, *Proc. 11^{th} RTA 2000 (Norwich)*, volume 1833 of *Lecture Notes in Computer Science*, pages 141–155. Springer, 2000.

[JM03] Felix Joachimski and Ralph Matthes. Short proofs of normalization for the simply-typed -calculus, permutative conversions and Gödel's t. *Archive for Mathematical Logic*, 2003. to appear.

[Joa01] Felix Joachimski. *Reduction Properties of ΠIE-Systems.* PhD thesis, LMU München, 2001. Available from `http://www.mathematik.uni-muenchen.de/~joachski`.

[Joa03] Felix Joachimski. Confluence of the coinductive lambda-calculus. *Theoretical Computer Science*, 2003. To appear, available from `http://www.mathematik.uni-muenchen.de/~joachski`.

[KMS75] G. Kreisel, G.E. Mints, and S.G. Simpson. The use of abstract language in elementary metamathematics: Some pedagogic examples. In R. Parikh, editor, *Logic Colloquium*, volume 453 of *Lecture Notes in Mathematics*, pages 38–131. Springer, 1975.

[Maa75] Wolfgang Maaß. Church-Rosser Theorem für λ-Kalküle mit unendlich langen Termen. In J. Diller and G.H. Müller, editors, *Proof Theory Symposion Kiel 1974*, volume 500 of *Lecture Notes in Mathematics*, pages 257–263. Springer, 1975.

[Min78] Grigori E. Mints. Finite investigations of transfinite derivations. *Journal of Soviet Mathematics*, 10:548–596, 1978. Translated from: Zap. Nauchn. Semin. LOMI 49 (1975). Cited after Grigori Mints. *Selected papers in Proof Theory.* Studies in Proof Theory. Bibliopolis, 1992.

[Rut00] Jan Rutten. Universal coalgebra: a theory of systems. *Theoretical Computer Science*, 249:3–80, 2000.

[Sch51] Kurt Schütte. Die unendliche Induktion in der Zahlentheorie. *Mathematische Annalen*, 122:369–389, 1951.

[Sch77] Kurt Schütte. *Proof Theory.* Springer, Berlin, 1977.

[Sch98] Helmut Schwichtenberg. Finite notations for infinite terms. *Annals of Pure and Applied Logic*, 94:201–222, 1998.

[TT97] Alastair Telford and David Turner. Ensuring Streams Flow. In Michael Johnson, editor, *Proc. 6th AMAST 1997 (Sydney)*, volume 1349 of *Lecture Notes in Computer Science*, pages 509–523. Springer, 1997.

[vR96] Femke van Raamsdonk. *Confluence and Normalisation for Higher-Order Rewriting.* Academisch Proefschrift (PhD thesis), Vrije Universiteit te Amsterdam, 1996.

[vRSSX99] Femke van Raamsdonk, Paula Severi, Morten Heine Sørensen, and Hongwei Xi. Perpetual reductions in lambda-calculus. *Information and Computation*, 149(2):173–225, 1999.

[Wei97] Andreas Weiermann. A proof of strongly uniform termination for Gödel's T by methods from local predicativity. *Archive for Mathematical Logic*, 36:445–460, 1997.