Arch. Math. Logic 42, 59–87 (2003)
Digital Object Identifier (DOI):
10.1007/s00153-002-0156-9

Mathematical Logic

Felix Joachimski · Ralph Matthes

Short proofs of normalization for the simply-typed λ-calculus, permutative conversions and Gödel's T

Received 12 October 1998
Published online: 19 December 2002 – © Springer-Verlag 2002

Abstract. Inductive characterizations of the sets of terms, the subset of strongly normalizing terms and normal forms are studied in order to reprove weak and strong normalization for the simply-typed λ-calculus and for an extension by sum types with permutative conversions. The analogous treatment of a new system with generalized applications inspired by generalized elimination rules in natural deduction, advocated by von Plato, shows the flexibility of the approach which does not use the strong computability/candidate style à la Tait and Girard. It is also shown that the extension of the system with permutative conversions by η-rules is still strongly normalizing, and likewise for an extension of the system of generalized applications by a rule of "immediate simplification". By introducing an infinitely branching inductive rule the method even extends to Gödel's T.

Introduction

Normalization proofs for the simply-typed λ-calculus enjoy continuous interest [Gog95, vRS95, vRSSX99] in the literature although the main results of weak and strong normalization have been established quite early by Turing (around 1941, published in [Gan80]) and Sanchis [San67].

This article employs a proof method that allows to show strong normalization for all typed terms without recourse to inclusive predicates such as strong computability [Tai67] or validity [Pra71] that are not formalizable in primitive recursive arithmetic. A simple induction on types verifies that *(strong) normalizability* is closed under application while closure under substitution has to be shown simultaneously.

Separating concerns. With the prospect of computer-checkable formalization and of proof-theoretic analyses of normalization proofs for more complicated calculi

F. Joachimski: Mathematisches Institut der Ludwig-Maximilians-Universität München, Germany. e-mail: felix@joachimski.de

R. Matthes: Institut für Informatik der Ludwig-Maximilians-Universität München, Germany. e-mail: matthes@informatik.uni-muenchen.de

Mathematics Subject Classification (2000): 03B40

Keywords or phrases: λ-calculus – Permutative/commuting conversions – Sum types – Generalized application – Gödel's T_{q^-} ω-rule – Inductive characterization

we identify independent steps in the proof. This is done by introducing strictly positive inductive definitions of the underlying concepts, thus abandoning geometric notions (such as positions in terms, reduction trees and reduction sequences) that would lead to ponderous arguments in the desired extensions.

Strong normalizability. Intuitively, a term r is *strongly normalizing* iff there are no infinite reduction sequences starting with r. However, the data type of reduction sequences is notoriously problematic for computer-assisted proof development. Following [Alt93] and the tradition in proof theory, we define strong normalizability of a term r (with respect to a binary relation $\rightarrow$) inductively by

$$\frac{\forall r'. r \rightarrow r' \Rightarrow r' \Downarrow}{r \Downarrow}$$

Thus, when checking strong normalizability of a term r it is sufficient to consider every one-step reduct of r instead of all possible (potentially infinite) reduction sequences.

Still, reduct analysis becomes increasingly annoying in normalization proofs for more and more complex systems. Therefore, we characterize strong normalizability via an inductive definition of a set SN given by syntax-directed Horn clauses, reducing the task of checking all one-step reducts to analysing no more than one *standard reduct* and some subterms. The proof of equivalence with the earlier definition is in fact the only place where reduct analysis has to be carried out.

For pure untyped λ-calculus such an inductive characterization has first been presented by van Raamsdonk/Severi [vRS95]. Among other goals, this article aims at extending the underlying notion to extensions of the base calculus by recursion, case operators and generalized eliminations (see below).[1]

Weak normalizability. A term r is *weakly normalizing* iff there is a finite reduction sequence starting with r and ending with a term which allows no further reduction. We give an inductive definition more suitable for a satisfying formalization. When checking weak normalization of a term r according to this definition one has to check some reduct of r. Unfortunately this reduct is not determined by r. Therefore, we also characterize weak normalizability via a syntax-directed inductive definition, thus embodying a reduction strategy that easily allows to read off which reduct to check.[2] Establishing the characterization this time amounts to proving standardization.

Building on these preliminary considerations the normalization proofs in sections 4–7 exclusively argue on the syntax-directed inductive definitions.

[1] Based on a draft version of the present article, an extension to a logical framework using dependent types has been carried out by Goguen [Gog99] whose typed operational semantics [Gog95] has been found independently of [vRS95]. Interestingly, the initial motivation for the work reported here was a kind of reverse engineering of Goguen's thesis [Gog94] for systems without dependent types.

[2] [vR96] stated and proved this characterization correct for pure untyped λ-calculus.

Notation for multiple eliminations. In the inductive characterization of strongly normalizing terms we recognize the *standard redex* of a term by displaying it in head form. This requires the introduction of a metasyntactic abbreviation: A list of terms $r_1, \dots, r_n$ is written $\vec{r}$ and can be used to denote iterated eliminations like in $x\vec{r}$ and $(\lambda xs)r\vec{r}$ (which stand for $xr_1 \dots r_n$ and $(\lambda xs)rr_1 \dots r_n$). Notice that those lists may actually be empty, leading to $x\vec{r} \equiv x$ (where $\equiv$ is syntactic equality).

It is crucial that we can carry this style of writing down multiple eliminations to more complex term systems as long as they enjoy an introduction/elimination dichotomy under the natural deduction (Curry-Howard) interpretation. This holds for ordinary term application, recursion on natural numbers (corresponding to N-elimination in natural deduction, i.e., induction on N) and case analysis ($+$-elimination), generalized applications à la von Plato as well as for many other systems not studied in this paper.[3] Thus we arrive at a more general notion of elimination rR of the term r that may stand for the application of r to R ($\to$-elimination with R a term), recursion rR with $R \equiv (s, t)$ (s the initial term, t the step term, usually written $\mathrm{Rec}\, s\, t\, r$), case analysis rR with $R \equiv (x.s, y.t)$ (commonly denoted by $\texttt{case}\ r$ $\texttt{of}\ \mathrm{inj}_0 x \Rightarrow s,\ \texttt{of}\ \mathrm{inj}_1\ y \Rightarrow t$) and generalized application rR with $R \equiv (s, x.t)$ (for $r \equiv y$ written as $t_x\{r, s\}$ in [Sch99]).

The vector notation turns out to replace concepts like branches and endsegments [Pra71] which have been important elements of normalization arguments in systems with permutative conversions.

Simply-typed λ-calculus. The basic system is dealt with in the first four sections: In section 1 untyped λ-calculus, function types, reduction and weak and strong normalizability are defined. In section 2 weak normalization is shown by a method relying on a proof by induction on normal forms. As we define weak normalizability from the set of normal forms this method is free from any notational overhead, i. e., any defined concept entering the proof is already needed for stating the result, viz. weak normalization. Section 3 contains the characterizations of weakly and strongly normalizing terms (also for the extension by the η-rule), section 4 the proof of strong normalization.

Permutative conversions. They—also called *commuting conversions*—recover the subformula property[4] for terms in normal form in a system with sum types and a case-construct. Difficulties arise with so-called *critical eliminations* $r(x.s, y.t) : \tau$ of a term r of type $\rho + \sigma$ where τ need neither be a subtype of ρ nor of σ. This has led to the invention of endsegments and validity predicates (see [Pra71] for a graphical and [vdP96] for a term-based definition) or improper reductions [Lei75].

[3] [Mat98] employs this approach in an analysis of extensions of Girard's system F by various forms of monotone inductive types. More recently, [Joa01] even gives a precise definition and analysis of the general notion of introduction/elimination systems which incorporate the underlying concept.

[4] "The type of any subterm of $r : \rho$ is subtype of either the type of a free variable or of ρ." See, e.g., [Hin97] for a good exposition.

In section 5 we use the vector notation together with an only marginal modification of the basic normalization argument and come up with a short and easily formalizable proof which even covers the η-reduction rules.

Generalized applications. They allow a further simplification of the grammar of normal forms for the λ-calculus by means of a generalized application rule that corresponds to the left introduction rule in sequent calculus. As in the case of sum types we have to add permutative conversions—now for function types—in order to retain the subformula property for normal terms. The methods of section 5 easily carry over to this system as shown in section 6, hence yielding strong normalization which appears to be a new result.

Gödel's T. Although the normalization proofs in the previous sections of this paper do make use of inductive definitions they could be formalized within primitive recursive arithmetic, since all inductive clauses have finitely many premises only (see [Tro73]). For reasons of proof-theoretical strength we cannot hope to extend the simple normalization proof to Gödel's T without major modifications since it is well-known that termination of T implies consistency of Peano arithmetic [Göd58] and that this fact is provable in primitive recursive arithmetic.

It will turn out that the inductive characterization SN of the strongly normalizing terms can be extended to T while the embedding of the set of all terms requires the introduction of an infinitely branching clause in an inductive definition of a subset $\Omega \subset \mathrm{SN}$ to cover recursion. A vaguely similar program has first been carried out for a combinatory version in [San67].[5] In section 7 we cover the full reduction relation of the λ-calculus with primitive recursion in finite types, profiting from our vector-based term display.

1. λ-calculus

1.1. Terms

(Raw) terms $r, s, t \in \Lambda$ are generated from variables x, y, z (of an infinite supply) by the grammar

$$\Lambda \ni r, s ::= x \mid rs \mid \lambda xr.$$

x is bound in λxr. Notationally, we let application associate to the left, writing $r\vec{s}$ for $rs_1 \dots s_n \equiv (\dots(rs_1)\dots s_n)$ with a possibly empty vector $\vec{s}$. As remarked in the introduction the use of lists to denote multiple eliminations is a metasyntactic device. Pointwise application of a function f or a predicate P to a list of terms $\vec{r}$ will be denoted by $f(\vec{r}\,)$ and $P(\vec{r}\,)$, respectively; the comma serves to denote concatenation and elongation: $r, \vec{r}, \vec{s}, s \equiv r, r_1, \dots, r_n, s_1, \dots s_m, s$.

We use the point notation $\lambda x.r$ to omit the parentheses in $\lambda x(r)$, where the range of the point extends as far as syntactically possible. The *simultaneous substitution* of terms $\vec{s}$ for $\vec{x}$ in r is defined as usual and written either $r_{\vec{x}}[\vec{s}\,]$ or $r[\vec{x} := \vec{s}\,]$. Terms which differ only in names of bound variables are identified, i.e., α-equal terms *are* equal.

[5] Other approaches can be found for instance in [Dil68], [How80] and [Sch93].

1.2. Inductive characterization and normal forms

The set of terms is inductively characterized by the following grammar —

$$\Lambda \ni r, s ::= x\vec{r} \mid \lambda xr \mid (\lambda xr)s\vec{s},$$

where the last form captures non-normal terms; we will see later that this presentation exhibits the leftmost outermost reducible expression (*redex*). Excluding this last clause we obtain the grammar of normal forms

$$\mathrm{NF} \ni r ::= x\vec{r} \mid \lambda xr.$$

1.3. Types and type assignment

ρ, σ, τ are generated from basic types ι by the grammar

$$\rho, \sigma ::= \iota \mid \rho \to \sigma.$$

Given a unique assignment $x : \rho$ of types to variable symbols such that for each type infinitely many variables exist, we can determine the typable terms and their unique type by the following rules:[6]

$$\frac{r : \rho \to \sigma \qquad s : \rho}{rs : \sigma} \qquad \frac{r : \sigma \qquad x : \rho}{\lambda xr : \rho \to \sigma}$$

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. $\Lambda^{\to}$ denotes the set of typable terms under a fixed type assignment for the variables.

1.4. Reduction

The reduction relation $\to \,:=\, \to_{\beta}$ is generated from the β-conversion rule $(\lambda xr)s \to r_x[s]$ by means of the *compatible closure*, extending conversion inductively by

$$r \to r' \implies \lambda xr \to \lambda xr', rs \to r's, sr \to sr'.$$

$\to^{+}$ is the transitive closure of $\to$ (defined inductively), $\to^{*}$ is the transitive, reflexive closure. The resulting reduction relation is also compatible in the sense that $r \to r'$ implies $s_x[r] \to^{*} s_x[r']$, and $s_x[r] \to s_x[r']$ if x occurs exactly once in s. It is *substitutive* insofar as $r \to r'$ implies $r_x[s] \to r'_x[s]$. That reduction preserves typability and types is a consequence of compatibility together with the fact that type-correct substitution respects types.[7]

We see how our inductive term characterization finds the leftmost outermost convertible expression[8] and that NF exactly captures the irreducible terms.

$\to_{\eta}$ is generated from the conversion rule $\lambda x.rx \to r$ (if x is not free in r) using the compatible closure. We write $\to_{\beta\eta}$ for $\to_{\beta} \cup \to_{\eta}$.

[6] Although our choice of typing discipline does shorten arguments considerably, it is not vital for the course of proofs, since we are only dealing with simple types here.

[7] We will not further comment on typability and type issues for the rest of this paper in order to keep the presentation short.

[8] See [Bar84] for a discussion of the mentioned concepts and various reduction strategies.

Notations. We define $r\downarrow$ (r is *weakly normalizing)* inductively by

$$\frac{r \in \mathrm{NF}}{r\downarrow} \qquad \frac{r \to r'\downarrow}{r\downarrow}$$

Thus $r\downarrow \Leftrightarrow \exists s \in \mathrm{NF}. r \to^* s$.

$r \Downarrow$ (r is *strongly normalizing*) is defined in the introduction. The set $\{r \mid r \Downarrow\}$ amounts to the well-founded part $\mathrm{WF}_{\to}$ of the reduction relation.[9] $r \Downarrow$ is equivalent to: *There is no infinite reduction sequence starting with r* [$\Rightarrow$: Induction on $r \Downarrow$. $\Leftarrow$ is an instance of bar induction]. We write $\#r$ for the finite height of the (finitely branching) reduction tree of strongly normalizing terms r, given recursively by $\#r := \sup\{\#r' + 1 \mid r \to r'\}$.

2. Weak normalization proof

2.1. The proof

In this section we assume that all mentioned terms are typable.

Lemma. $r, s^\rho \in NF \implies (i)\ rs\downarrow,\ (ii)\ r_x[s]\downarrow.$

Proof. By simultaneous induction on ρ, side induction on $r \in \mathrm{NF}$.

$y\vec{r}$. Assume $y\vec{r} \in \mathrm{NF}$ has been obtained from $\vec{r} \in \mathrm{NF}$. (i). $y\vec{r}s \in \mathrm{NF}$ follows immediately. (ii). Side induction hypothesis (ii) shows $\vec{r}_x[s]\downarrow$; let $\vec{t}$ be the respective normal forms. If $x \not\equiv y$ then $(y\vec{r}\,)_x[s] = y\vec{r}_x[s]\downarrow$. Otherwise

$$(y\vec{r}\,)_x[s] \equiv (x\vec{r}\,)_x[s] = s\vec{r}_x[s] \to^* s\vec{t}.$$

Thus for $s\vec{r}_x[s]\downarrow$ it suffices to show $s\vec{t}\downarrow$. For this we use an additional induction on the length of $\vec{t}$. If this vector is empty $s\vec{t} \equiv s \in \mathrm{NF}$ by assumption. Otherwise $\vec{t} \equiv \vec{t'}, t$ and by induction hypothesis we have $s\vec{t'}\downarrow$ with normal form s'. Since $s\vec{t'}$ is applied to t we know that the type of t is a subtype of the type ρ of s. Hence the main induction hypothesis (i) yields

[9] Let $\succ$ be a binary relation on the fixed set M.

$$\begin{array}{lll} \mathrm{Prog}_\succ(X) & :\Leftrightarrow \forall x \in M.(\forall y \prec x. y \in X) \Rightarrow x \in X, & (X \text{ is } \succ\textit{-progressive}) \\ \mathrm{WF}_\succ & := \bigcap\{X \subseteq M \mid \mathrm{Prog}_\succ(X)\}. & \end{array}$$

- The definition of the *well-founded part* $\mathrm{WF}_\succ$ yields the schema $\mathrm{Prog}_\succ(X) \implies \mathrm{WF}_\succ \subseteq X$ of *accessible part induction.*
- $\mathrm{WF}_\succ$ is $\succ$-progressive itself.
- Definition by recursion on $\mathrm{WF}_\succ$ is admissible, since $\mathrm{WF}_\succ$ is an example of a deterministic inductive definition.
- If $\succ \subseteq \succ'$ then $\mathrm{Prog}_\succ(X)$ implies $\mathrm{Prog}_{\succ'}(X)$, hence $\mathrm{WF}_{\succ'} \subseteq \mathrm{WF}_\succ$, i.e., $\lambda\succ.\mathrm{WF}_\succ$ is *antitone* with respect to $\subseteq$.
- $\mathrm{WF}_\succ = \mathrm{WF}_{\succ^+}$ where $\succ^+$ is the transitive closure of $\succ$. [For the interesting part $\mathrm{WF}_\succ \subseteq \mathrm{WF}_{\succ^+}$ one has to show that $\mathrm{WF}_{\succ^+}$ is $\succ$-progressive. $\supseteq$ uses antitonicity.] This means that "course-of-generation-induction" is a derived induction principle, the use of which will be announced as induction on $\mathrm{WF}_{\succ^+}$.

$s't \downarrow$, with normal form s''. Putting the reductions together we get (with the help of compatibility)

$$s\vec{t} \equiv s\vec{t'}t \to^* s't \to^* s'' \in \mathrm{NF}.$$

λyr. (i). $\lambda yr \in \mathrm{NF}$ is derived from $r \in \mathrm{NF}$. Hence the side induction hypothesis (ii) for r proves $r_y[s] \downarrow$. Consequently $(\lambda yr)s \to r_y[s]$ also normalizes.
(ii). Without loss of generality $y \not\equiv x$ and y not free in s[10], so $(\lambda yr)_x[s] = \lambda yr_x[s]$. By side induction hypothesis (ii) $r_x[s] \downarrow$, thus $\lambda yr_x[s] \downarrow$. □

Theorem. $r^\rho \downarrow$.

Proof. By induction on r. $x \in \mathrm{NF}$ by definition. $rs \downarrow$ by the lemma (part (i)), using the induction hypothesis for r and s. $\lambda xr \downarrow$, since by induction hypothesis $r \downarrow$. □

2.2. *Extracted program*

By formalizing this constructive normalization proof in the style of [Ber93] we obtain

- the algorithmic content $\mathrm{nf} : \Lambda^{\to} \to \mathrm{NF}$ of the theorem (defined by recursion on the term structure):

$$\begin{aligned} \mathrm{nf}(x) &:= x \\ \mathrm{nf}(\lambda xr) &:= \lambda x\ \mathrm{nf}(r) \\ \mathrm{nf}(rs) &:= \mathrm{app}(\mathrm{nf}(r), \mathrm{nf}(s)) \end{aligned}$$

- the algorithms $\mathrm{app} : \mathrm{NF} \times \mathrm{NF} \to \mathrm{NF}$, $\mathrm{subst}_x : \mathrm{NF} \times \mathrm{NF} \to \mathrm{NF}$ (with auxiliary function $\mathrm{app}_s : \mathrm{NF}^* \to \mathrm{NF}$ for $s \in \mathrm{NF}$ where NF^* denotes the set of finite lists over NF and ε is the empty list), extracted from the proofs of part (i) and (ii) of the lemma (defined by recursion corresponding to the induction principles used):

$$\begin{aligned} &\mathrm{app}(y\vec{r}, s) := y\vec{r}s \\ &\mathrm{app}(\lambda yr, s) := \mathrm{subst}_y(r, s) \\[1ex] &\mathrm{subst}_x(y\vec{r}, s) := \text{if } x \equiv y \text{ then } \mathrm{app}_s(\mathrm{subst}_x(\vec{r}, s)) \text{ else } y\ \mathrm{subst}_x(\vec{r}, s) \\ &\quad \mathrm{app}_s(\varepsilon) := s \\ &\quad \mathrm{app}_s(\vec{t}, t) := \mathrm{app}(\mathrm{app}_s(\vec{t}), t) \\ &\mathrm{subst}_x(\lambda yr, s) := \lambda y\ \mathrm{subst}_x(r, s) \end{aligned}$$

2.3. *A reformulation*

First notice that

$$(*) \qquad r \in \mathrm{NF} \Longrightarrow rx \downarrow$$

[Case λxr:[11] $(\lambda xr)x \to r \in \mathrm{NF}$. Case $y\vec{r}$: $y\vec{r}x \in \mathrm{NF}$ if $\vec{r} \in \mathrm{NF}$]. This fact can be used to merge the two assertions of the lemma into one that speaks about substitution only, allowing a shorter proof that does no more involve an induction on the length of the application in the case of (ii) $y\vec{r}$. We are grateful to René David for hinting at this trick which he found and showed to us for the strong normalization

[10] thanks to the identification of α-equal terms

[11] This is a rather subtle application of bound variable renaming. It is, of course, possible to make it explicit.

case (see section 4) after our presentation of a very early version of the present paper.[12]

Lemma. $r, s^\rho \in NF \Longrightarrow r_x[s]\downarrow$.

Proof. By induction on ρ, side induction on $r \in$ NF.

Case $y\vec{r}$. The proof is trivial if $\vec{r}$ is empty or $x \not\equiv y$. Otherwise $y\vec{r} = (xt^{\rho_0})^{\rho_1}\vec{t}$ with $\rho = \rho_0 \to \rho_1$.

- By the side induction hypothesis $t_x[s]\downarrow, \vec{t}_x[s]\downarrow$ with normal forms $t', \vec{t}'$. Hence $z_1\vec{t}' \in$ NF for a new variable z_1 of type ρ_1.
- $(*)$ shows $sz_0\downarrow$ for any new variable z_0 of type ρ_0, let s' be a[13] normal form. By substitutivity
$$st' = (sz_0)_{z_0}[t'] \to^* s'_{z_0}[t'].$$
By induction hypothesis at ρ_0 (which is a subtype of ρ) we get $s'_{z_0}[t']\downarrow$ with normal form — say — s_0.

Together this allows to apply the induction hypothesis for ρ_1 to obtain $(z_1\vec{t}')_{z_1}[s_0]\downarrow$ and this provides a normal form of the term in question:
$$(xt\vec{t})_x[s] = st_x[s]\vec{t}_x[s] \to^* st'\vec{t}' \to^* s_0\vec{t}'\downarrow.$$

Case λyr. $(\lambda yr)_x[s] = \lambda yr_x[s]$, so the side induction hypothesis proves the claim. □

Corollary. $r, s \in NF \Longrightarrow rs\downarrow$.

Proof. Use the lemma for xs and r with new x. □

The extracted algorithms subst_x and app differ from the previously found programs and use the constructive content $\text{app}^x : \text{NF} \to \text{NF}$ of the proof of $(*)$.

```
app^x(λxr) := r [14]
app^x(y r⃗) := y r⃗ x

subst_x(λyr, s) := λy subst_x(r, s)
subst_x(y, s) := if x ≡ y then s else y
subst_x(y t t⃗, s) := if x ≡ y
        then subst_{z1}(z1 subst_x(t⃗, s), subst_{z0}(app^{z0}(s), subst_x(t, s)))
                (z0 not free in s, z1 not free in t⃗, s)
        else y subst_x(t, s) subst_x(t⃗, s)

app(r, s) := subst_x(xs, r),     (x not free in s)
```

$$\begin{aligned}
\text{app}^x(\lambda xr) &:= r\ [14]\\
\text{app}^x(y\vec{r}) &:= y\vec{r}x
\end{aligned}$$

$$\begin{aligned}
\text{subst}_x(\lambda yr, s) &:= \lambda y\ \text{subst}_x(r, s)\\
\text{subst}_x(y, s) &:= \text{if } x \equiv y \text{ then } s \text{ else } y\\
\text{subst}_x(yt\vec{t}, s) &:= \text{if } x \equiv y\\
&\quad\ \text{then } \text{subst}_{z_1}(z_1\text{subst}_x(\vec{t}, s), \text{subst}_{z_0}(\text{app}^{z_0}(s), \text{subst}_x(t, s)))\\
&\qquad\qquad (z_0 \text{ not free in } s,\ z_1 \text{ not free in } \vec{t}, s)\\
&\quad\ \text{else } y\ \text{subst}_x(t, s)\text{subst}_x(\vec{t}, s)\\
\text{app}(r, s) &:= \text{subst}_x(xs, r), \qquad (x \text{ not free in } s)
\end{aligned}$$

[12] In [Dav01], this idea is made fruitful for the uniform proof of several known and also new characterizations of terms typable in systems of intersection types by normalization properties.

[13] Confluence ensures that normal forms are unique but this is not required throughout the paper.

[14] If we used explicit renaming of bound variables this clause would read $\text{app}^x(\lambda yr) := r_y[x]$.

3. The sets WN and SN

In the proof of Lemma 2.1 we used three closure properties of the set $\{r \mid r \downarrow\}$.

$$\begin{aligned} \vec{r}\downarrow &\Longrightarrow x\vec{r}\downarrow, \\ r\downarrow &\Longrightarrow \lambda xr\downarrow, \\ r_x[s]\vec{s}\downarrow &\Longrightarrow (\lambda xr)s\vec{s}\downarrow . \end{aligned}$$

We now use these clauses to generate the sets WN and SN inductively which will turn out to capture the concepts *weakly* and *strongly normalizing* exactly, even in the context of untyped λ-calculus. The characterization of WN and SN with respect to the intuitive notions of weak and strong normalizability can be found in [vRS95].[15]

3.1. Definition

The sets SN ⊂ WN ⊂ Λ are defined inductively by the following rules.

$$\frac{\vec{r}\in \mathrm{WN}}{x\vec{r}\in \mathrm{WN}}(\mathrm{Var}) \qquad \frac{r\in \mathrm{WN}}{\lambda xr\in \mathrm{WN}}(\lambda) \qquad \frac{r_x[s]\vec{s}\in \mathrm{WN}}{(\lambda xr)s\vec{s}\in \mathrm{WN}}(\beta)$$

$$\frac{\vec{r}\in \mathrm{SN}}{x\vec{r}\in \mathrm{SN}}(\mathrm{Var}) \qquad \frac{r\in \mathrm{SN}}{\lambda xr\in \mathrm{SN}}(\lambda) \qquad \frac{r_x[s]\vec{s}\in \mathrm{SN} \qquad s\in \mathrm{SN}}{(\lambda xr)s\vec{s}\in \mathrm{SN}}(\beta)$$

Remark. The derived term of each rule has a different form — in fact, there is a one-to-one correspondence between the rules and the grammar rules in 1.2 — and therefore the sets WN and SN are generated freely, rendering recursion admissible. Furthermore both sets contain NF by definition.

Using the abbreviation $\mho := (\lambda x.xx)(\lambda x.xx) \in \Lambda \setminus \mathrm{WN}$ we see that $(\lambda yz)\mho \in \mathrm{WN} \setminus \mathrm{SN}$.

3.2. Weak normalization for WN

$r \in \mathrm{WN}$ implies $r\downarrow$, since the defining rules are closure properties of $\{r \mid r\downarrow\}$. But the converse direction also holds,[16] so we have $r \in \mathrm{WN} \Longleftrightarrow r\downarrow$. The normal form $\downarrow(r)$ of a term $r \in \mathrm{WN}$ can be defined by recursion as follows.[17]

[15] See the introduction for the reasons why we do not simply refer to those results and the equivalence of the intuitive definitions with $r\downarrow$ and $r\Downarrow$.

[16] This means that the reduction strategy that underlies the definition of WN actually is the standard reduction strategy guaranteed to exist by the standardization theorem (see, e.g., [Dav95] for a short exposition).

[17] Goguen's typed operational semantics [Gog95], which inspired our work, can be understood as a combination of SN, NF and the normal form function $\downarrow$ into one inductive definition.

$$\begin{aligned}
\downarrow &: \mathrm{WN} \longrightarrow \mathrm{NF},\\
\downarrow(x\vec{r}\,) &:= x \downarrow(\vec{r}\,),\\
\downarrow(\lambda x r) &:= \lambda x \downarrow(r),\\
\downarrow((\lambda x r) s\vec{s}\,) &:= \downarrow(r_x[s]\vec{s}\,).
\end{aligned}$$

3.3. Soundness

Lemma. *SN* $\subseteq$ *WF*$_{\rightarrow}$.

Proof. By induction on SN, i.e., by showing that the set of strongly normalizing terms is closed under the rules defining SN.

(Var). Assume $\vec{r} \Downarrow$. Since each reduction on $x\vec{r}$ has to take place in one of the $\vec{r}$, we also get $x\vec{r} \Downarrow$.

(λ). Each reduction on $\lambda x r$ actually occurs in r, so the hypothesis $r \Downarrow$ yields $\lambda x r \Downarrow$.

(β). We show $(\lambda x r)s\vec{s} \Downarrow$ by main induction on $s \Downarrow$ and side induction on $r_x[s]\vec{s} \Downarrow$.[18] We have to prove $t \Downarrow$ for any immediate reduct t. The following reductions are possible.

$(\lambda x r)s\vec{s} \rightarrow (\lambda x r')s\vec{s}$. Then $r_x[s]\vec{s} \rightarrow r'_x[s]\vec{s}$ by substitutivity, hence by side induction hypothesis $(\lambda x r')s\vec{s} \Downarrow$.

$(\lambda x r)s\vec{s} \rightarrow (\lambda x r)s'\vec{s}$. Then $r_x[s]\vec{s} \rightarrow^* r_x[s']\vec{s}$ by compatibility, hence also $r_x[s']\vec{s} \Downarrow$. The main induction hypothesis yields $(\lambda x r)s'\vec{s} \Downarrow$.

$(\lambda x r)s\vec{s} \rightarrow (\lambda x r)s\vec{s'}$. Then $r_x[s]\vec{s} \rightarrow r_x[s]\vec{s'}$, hence by side induction hypothesis $(\lambda x r)s\vec{s'} \Downarrow$.

$(\lambda x r)s\vec{s} \rightarrow r_x[s]\vec{s} \Downarrow$ by assumption. □

Along the lines of the proof one can verify the following inductive bounds on the height $\#r$ of the reduction tree for terms $r \in$ SN:

$$\#x\vec{r} \leq \sum \#\vec{r}, \quad \#\lambda x r \leq \#r^{19} \quad \text{and} \quad \#(\lambda x r)s\vec{s} \leq 1 + \#r_x[s]\vec{s} + \#s.$$

[18] This argument is crucial to the proof of strong normalization and to our knowledge it is an unavoidable part of any strong normalization proof amenable to a faithful formalization. Most often, however, it is hidden in an argument via infinite reduction sequences. In order to clarify the intricate structure of the argument we spell out the nesting of inductions precisely. Main induction on $s \Downarrow$ amounts to showing $\rightarrow$-progressivity of the set

$$M := \{s \mid \forall r, \vec{s}. r_x[s]\vec{s} \Downarrow \Rightarrow (\lambda x r)s\vec{s} \Downarrow\}.$$

So assume that any one-step-reduct of s is in M. For $s \in M$ we do side induction on $r_x[s]\vec{s} \Downarrow$, i.e., we show $\rightarrow$-progressivity of the set

$$N := \{t' \mid t' \Downarrow \wedge \forall r, \vec{s}. t' = r_x[s]\vec{s} \Rightarrow (\lambda x r)s\vec{s} \Downarrow\}.$$

So assume t' is a term and each immediate reduct is in N. We have to show $t' \in N$. $t' \Downarrow$ follows from the definition of $\Downarrow$, since $N \subseteq \mathrm{WF}_{\rightarrow}$. If t' has the form $r_x[s]\vec{s}$ we have to show $(\lambda x r)s\vec{s} \Downarrow$, so we prove $t \Downarrow$ for every one-step reduct t. Henceforward, each clause of the proof in the main text can be recast using the pending assumptions.

[19] Clearly, the first two inequalities even hold with $=$.

3.4. Completeness

For the purpose of the normalization proof it is not necessary to know that SN captures all strongly normalizing terms. Nevertheless the result is interesting in its own right.

Lemma. *$WF_{\to} \subseteq SN$.*

The proof uses nested inductions on $\mathrm{WF}_{\to}$ and the term structure. In fact, the lemma is an instance of the following proposition, the generality of which will pay off in section 5.

Proposition. *Let $\to$ be a binary relation on the set $WF_{\rhd}$ with*

$$\forall r\, \forall r'\, \forall r''.r \rhd r' \to r'' \Rightarrow \exists r'''.r \to r''' \rhd r'' \qquad \textit{(commutation).}$$

Then $WF_{\to} \subseteq WF_{\rhd\cup\to}$.

Proof. By (accessible part) induction on $\to$. With $\succ := \rhd \cup \to$ our goal is $\mathrm{Prog}_{\to}(\mathrm{WF}_{\succ})$, which reads $\forall r.(\forall s \leftarrow r.s \in \mathrm{WF}_{\succ}) \Rightarrow r \in \mathrm{WF}_{\succ}$. Using induction on $\rhd$ we need to prove

$$\mathrm{Prog}_{\rhd}(\lambda r.(\underbrace{\forall s \leftarrow r.s \in \mathrm{WF}_{\succ}}_{\equiv:\Delta(r)}) \Rightarrow r \in \mathrm{WF}_{\succ})$$
$$\equiv \forall r.(\forall t \lhd r.\Delta(t) \Rightarrow t \in \mathrm{WF}_{\succ}) \wedge \Delta(r) \Rightarrow r \in \mathrm{WF}_{\succ}.$$

So assume r with

(1) $\forall t \lhd r.\Delta(t) \Rightarrow t \in \mathrm{WF}_{\succ}$ and
(2) $\Delta(r) \equiv \forall s \leftarrow r.s \in \mathrm{WF}_{\succ}$.

We need to show $r \in \mathrm{WF}_{\succ}$. Since the well-founded part $\mathrm{WF}_{\succ}$ is $\succ$-progressive it suffices to show $\forall s \prec r.s \in \mathrm{WF}_{\succ}$. So let $r \succ s$.

Case $r \to s$. Then $s \in \mathrm{WF}_{\succ}$ holds by (2).
Case $r \rhd s$. Using (1) it suffices to know $\Delta(s)$, i.e., $\forall t \leftarrow s.t \in \mathrm{WF}_{\succ}$. So assume $r \rhd s \to t$. Then by commutation there exists an r' with $r \to r' \rhd t$, so by (2) also $r' \in \mathrm{WF}_{\succ}$, hence $t \in \mathrm{WF}_{\succ}$ as $r' \succ t$. □

To get the lemma, we first apply the proposition to $\to$ and a restricted subterm relation $\rhd$ given by the clauses

$$x\vec{r} \rhd r_k, \quad \lambda x r \rhd r, \quad \text{and} \quad (\lambda x r)s\vec{s} \rhd s.$$

Clearly, $\mathrm{WF}_{\rhd} = \Lambda$. Commutation is shown by simple case analysis.

$$\lambda x r \rhd r \to r' \Longrightarrow \lambda x r \to \lambda x r' \rhd r',$$
$$x\vec{r} \rhd r_k \to r'_k \Longrightarrow x\vec{r} \to x\vec{r'} \rhd r'_k,$$
$$(\lambda x r)s\vec{s} \rhd s \to s' \Longrightarrow (\lambda x r)s\vec{s} \to (\lambda x r)s'\vec{s} \rhd s'.$$

The proposition yields $\mathrm{WF}_{\to} \subseteq \mathrm{WF}_{\rhd\cup\to}$. The latter is a subset of SN by antitonicity of $\lambda\succ.\mathrm{WF}_{\succ}$, since SN is generated as $\mathrm{WF}_{\sqsupset}$ with $\sqsupset$ extending $\rhd$ by head-conversion

$$(\lambda x r)s\vec{s} \sqsupset r_x[s]\vec{s}.$$

3.5. On η

The set SN also captures exactly the notion of *strongly normalizing* for $\to_{\beta\eta}$. For $\mathrm{WF}_{\to_{\beta\eta}} \subseteq$ SN we apply antitonicity of $\lambda\succ.\mathrm{WF}_{\succ}$. For the converse inclusion, the proof of Lemma 3.3 remains largely unaltered due to the fact that η-reductions on the left side of an application (as in $(\lambda x.rx)s \to_\eta rs$) are also β-reductions and thus need not be considered. Furthermore η-reductions on the right side of an application (i.e., $r\lambda x.sx \to_\eta rs$) are always covered by the induction hypothesis. So we only have to deal with the new case of a head-η-reduction $\lambda x.sx \to s$ in part (λ) of the proof when showing that $\lambda x.sx \Downarrow$ follows from $sx \Downarrow$, but clearly $s \Downarrow$, since the set $\{r \mid r \Downarrow\} = \mathrm{WF}_{\to_{\beta\eta}}$ of strongly normalizing terms is closed under subterms.

It has been shown elsewhere [AJ01] that strong normalization (and confluence) of combined β-reduction and η-expansion follows from the same property of β-reduction alone. As a consequence SN also exactly characterizes the set of strongly normalizing terms with respect to $\to_{\beta\eta\Uparrow}$ (defined loc. cit.).

3.6. The SN-method [Mat98]

Having the characterization of strongly normalizing terms via SN at hand, one can rearrange traditional strong normalization proofs, e.g., by the method of logical predicates, such that they do not refer to reduction. Using suitable extensions of the vector notation, SN can also be defined for system F and extensions such as by monotone inductive types. The *saturated sets* variant [Tai75] of Girard's normalization proof may be adapted, with the notion of saturatedness naturally arising from the definition of SN. This is also true for systems with permutative conversions as shown, e. g., in [Mat00].

4. Strong normalization proof

In this section we lift the proof of Lemma 2.1 to strong normalization, using the SN-method explained above. We assume throughout that all displayed terms are typable.

Lemma. $r \in SN, s^\rho \in SN \implies (i)\ rs \in SN,\ (ii)\ r_x[s] \in SN.$

Proof. By simultaneous induction on ρ and side induction on the generation of $r \in$ SN. In essence the proof is parallel to that of Lemma 2.1 but the use of SN shortens arguments considerably.

(Var). $r \equiv y\vec{r} \in$ SN is generated from $\vec{r} \in$ SN. (i). $\vec{r} \in$ SN together with $s \in$ SN shows $y\vec{r}s \in$ SN, using (Var). (ii). Side induction hypothesis (ii) yields $r_k[x := s] \in$ SN for each r_k in $\vec{r}$. Now if $y \not\equiv x$ we get $y\vec{r}_x[s] \in$ SN at once. Otherwise we need $s\vec{r}_x[s] \in$ SN, but this follows by multiple applications of the main induction hypothesis for (i), since each of the arguments in the nested application has a subtype of ρ.[20]

[20] More precisely, this argument involves an additional induction on the length of the list $\vec{r}$, analogous to the proof of Lemma 2.1.

(λ). $\lambda yr \in \mathrm{SN}$ is generated from $r \in \mathrm{SN}$. (i). For $(\lambda yr)s \in \mathrm{SN}$ it suffices to show $r_y[s] \in \mathrm{SN}$. But this is exactly the side induction hypothesis for (ii). (ii). Side induction hypothesis (ii) yields $r_x[s] \in \mathrm{SN}$ and consequently $\lambda y r_x[s] \in \mathrm{SN}$.

(β). $(\lambda yr)t\vec{t} \in \mathrm{SN}$ is obtained from $r_y[t]\vec{t} \in \mathrm{SN}$ and $t \in \mathrm{SN}$. Therefore both side induction hypotheses apply and yield (i) $r_y[t]\vec{t}s \in \mathrm{SN}$ and (ii) $(r_y[t]\vec{t})_x[s], t_x[s] \in \mathrm{SN}$, respectively. From these we obtain the assertions $(\lambda yr)t\vec{t}s \in \mathrm{SN}$ (i) and $((\lambda yr)t\vec{t})_x[s] \in \mathrm{SN}$ (ii) by definition of SN. □

Remark. A proof of the statement of the lemma with SN replaced by WN is only marginally simpler.

Theorem. *All terms are strongly normalizable.*

Proof. Show $r \in \mathrm{SN}$ by induction on the structure of r, using the lemma. Lemma 3.3 proves strong normalizability of r. □

Again (compare with 2.3), we can merge parts (i) and (ii) of the lemma into closure under substitution using the

Proposition. $r \in SN \Longrightarrow rx \in SN.$[21]

Proof. By induction on SN. □

Lemma. $r, s^\rho \in SN \Longrightarrow r_x[s] \in SN.$

Proof. By induction on ρ, side induction on $r \in \mathrm{SN}$. We only mention the variable case with a non-trivial elimination $(xt^{\rho_0})^{\rho_1}\vec{t}$ on the same variable x that is substituted by s. Then $\rho = \rho_0 \to \rho_1$.

- By the side induction hypothesis we have $t_x[s], \vec{t}_x[s] \in \mathrm{SN}$. Consequently also $z\vec{t}_x[s] \in \mathrm{SN}$ for a fresh variable $z : \rho_1$.
- By the proposition we have $sy^{\rho_0} \in \mathrm{SN}$ for a new y, hence by induction hypothesis for ρ_0 also $st_x[s] \in \mathrm{SN}$.
- The induction hypothesis for ρ_1 shows $(xt\vec{t})_x[s] = (z\vec{t}_x[s])_z[st_x[s]] \in \mathrm{SN}$.

□

As a corollary we get closure of SN under application, again.

5. Extension to permutative conversions

We proceed to a typed calculus with sums and permutative conversions similar to those being studied in [Pra71], [Lei75] or [GLT89]. In contrast to their expositions we do not consider product types or other first-order extensions, although the proof method copes with them, too.

5.1. (Raw) terms

Let always $i \in \{0, 1\}$.

$$r, s, t ::= x \mid \lambda yr \mid rs \mid \mathrm{inj}_i r \mid s(x_0.t_0, x_1.t_1).$$

The variables y, x_0, x_1 get bound in r, t_0, t_1, respectively.

[21] Induction on $rx \in \mathrm{SN}$ would show the converse direction.

5.2. Types and type assignment

The type grammar is extended to sums $\rho + \sigma$. In order to get unique types for typable terms we add a type subscript to the injection (writing $\mathrm{inj}_{i,\rho}$) that will, however, be omitted.

$$\frac{r : \rho_i}{\mathrm{inj}_{i,\rho_{1-i}} r : \rho_0 + \rho_1} \qquad \frac{r : \rho_0 + \rho_1 \qquad s_0 : \sigma \qquad s_1 : \sigma}{r(x_0^{\rho_0}.s_0, x_1^{\rho_1}.s_1) : \sigma}$$

(The type superscripts of variables in the second rule indicate the type assignments for those variables. In fact, the assignments are premises of the rule.) Note that typing rules out pathological terms like $(\lambda x r)(x.s, y.t)$ or $(\mathrm{inj}_i r)s$.

In the rest of this section we require all mentioned terms to be typable.

5.3. Inductive characterization

We use capital letters R, S, T for *eliminations*[22], which are either terms or sum-eliminations $(x_0.s_0, x_1.s_1)$:

$$R ::= r \mid (x_0.s_0, x_1.s_1).$$

The terms r, s_0 and s_1 are named *elimination terms* in this definition. Eliminations of the form $(x_0.s_0, x_1.s_1)$ are called *critical*. We write $(x_0.s_0, x_1.s_1)_x[t]$ for the capture-free substitution $(x_0.s_0[x := t], x_1.s_1[x := t])$.

Using a square bracket notation [...] for optional syntax elements, we can inductively characterize the set of typed terms by

$$\begin{aligned}\Lambda^{\to +} \ni r, s ::= {} & x\vec{r}\,[R] \mid \lambda x r \mid \mathrm{inj}_i r \mid \\ & x\vec{r}(x_0.s_0, x_1.s_1)S\vec{S} \mid (\lambda x r)s\vec{S} \mid (\mathrm{inj}_i)(x_0.s_0, x_1.s_1)\vec{R}.\end{aligned}$$

5.4. Normal forms

The last three clauses in the inductive term characterization are not considered normal. This is intuitive for the β-redices $(\lambda x r)s\vec{S}$ and $(\mathrm{inj}_i r)(x_0.s_0, x_1.s_1)\vec{S}$. The term $x\vec{r}(x_0.s_0, x_1.s_1)S\vec{S}$ need not fulfil the subformula property in general.

$$\mathrm{NF} \ni r ::= x\vec{r}\,[(x_0.s_0, x_1.s_1)] \mid \lambda x r \mid \mathrm{inj}_i r.$$

Notice how the inductive term characterization uniquely determines the canonical redex of non-normal terms.

5.5. Reduction

Redices are removed by means of β-conversion

$$(\lambda x r)s \to_{\beta_{\to}} r_x[s], \qquad (\mathrm{inj}_i r)(x_0.s_0, x_1.s_1) \to_{\beta_+} s_i[x_i := r],$$

[22] Note the ambiguity: R as well as rR are called eliminations (cf. the introduction).

and permutative conversions

$$r(x_0.s_0, x_1.s_1)S \to_\pi r(x_0.s_0S, x_1.s_1S).$$[23]

The *compatible closure* $\to$ of these conversions is given by

$$r \to r' \implies sr \to sr', rS \to r'S, \lambda xr \to \lambda xr', \mathrm{inj}_i r \to \mathrm{inj}_i r',$$
$$s(x.r, y.t) \to s(x.r', y.t), s(y.t, x.r) \to s(y.t, x.r').$$

5.6. Definition

The set SN is generated inductively by the following rules where $(x.r, y.t) \in \mathrm{SN}$ is an abbreviation for $r, t \in \mathrm{SN}$.

$$\frac{\vec{r}\,[,R] \in \mathrm{SN}}{x\vec{r}\,[R] \in \mathrm{SN}}(\mathrm{Var}) \qquad \frac{r \in \mathrm{SN}}{\lambda xr \in \mathrm{SN}}(\lambda)$$

$$\frac{x\vec{r}(x_0.s_0S, x_1.s_1S)\vec{S} \in \mathrm{SN}}{x\vec{r}(x_0.s_0, x_1.s_1)S\vec{S} \in \mathrm{SN}}(\mathrm{Var}_\pi)^{24} \qquad \frac{r \in \mathrm{SN}}{\mathrm{inj}_i r \in \mathrm{SN}}(\mathrm{inj})$$

$$\frac{r_x[s]\vec{S} \in \mathrm{SN} \qquad s \in \mathrm{SN}}{(\lambda xr)s\vec{S} \in \mathrm{SN}}(\beta_\to)$$

$$\frac{s_i[x_i := r]\vec{S} \in \mathrm{SN} \qquad s_{1-i}\vec{S} \in \mathrm{SN} \qquad r \in \mathrm{SN}}{(\mathrm{inj}_i r)(x_0.s_0, x_1.s_1)\vec{S} \in \mathrm{SN}}(\beta_+)$$

Remark. $x\vec{R}$ is necessarily generated from an $x\vec{r}\,[R]$ where a potentially critical elimination R occurs outermost, if at all. Furthermore $x\vec{r}\,[R]$ can be obtained from $x\vec{R}$ by permutative conversions.

It is also worth mentioning that the definition of SN incorporates permutations for variable eliminations only. The following lemma shows that this is sufficient to capture all strongly normalizing terms.

Lemma. *SN* $=$ *WF*$_\to$.

Proof. "$\subseteq$" (soundness): We again have to show that $\mathrm{WF}_\to = \{r \mid r \Downarrow\}$ has the closure properties which define SN. For (Var) and (λ) the proof remains unchanged, (inj_i) is as trivial as (λ). The proof for ($\beta_\to$) is verbosely the same as that in section 3, replacing $\vec{s}$ by $\vec{S}$. Notice that now permutative conversions may affect the length of the list $\vec{S}$.

[23] Strictly speaking, we require x_0, x_1 not to be free in S. Otherwise the rule would not comply with α-equality.

[24] This clause is subject to the same proviso as the rule of permutative conversion.

(Var_π). Show $x\vec{r}(x_0.s_0S, x_1.s_1S)\vec{S} \Downarrow \Longrightarrow x\vec{r}(x_0.s_0, x_1.s_1)S\vec{S} \Downarrow$ by induction on $x\vec{r}(x_0.s_0S, x_1.s_1S)\vec{S} \in \mathrm{WF}_{\to^+}$. We have to show $t \Downarrow$ for every reduct t of $x\vec{r}(x_0.s_0, x_1.s_1)S\vec{S}$. The only interesting case arises with $\vec{S} \equiv T\vec{T}$ and a permutation of $S \equiv (y_0.t_0, y_1.t_1)$ with T leading to

$$t = x\vec{r}(x_0.s_0, x_1.s_1)(y_0.t_0T, y_1.t_1T)\vec{T}.$$

$t \Downarrow$ follows by induction hypothesis as

$$x\vec{r}(x_0.s_0S, x_1.s_1S)\vec{S} \equiv x\vec{r}(x_0.s_0(y_0.t_0, y_1.t_1), x_1.s_1(y_0.t_0, y_1.t_1))T\vec{T}$$

reduces with three permutative reduction steps (hence once $\to^+$) to

$$x\vec{r}(x_0.s_0(y_0.t_0T, y_1.t_1T), x_1.s_1(y_0.t_0T, y_1.t_1T))\vec{T}.$$

(β_+). Show that $s_i[x_i := r]\vec{S} \Downarrow, s_{1-i}\vec{S} \Downarrow$ and $r \Downarrow$ imply $(\mathrm{inj}_i r)(x_0.s_0, x_1.s_1)\vec{S} \Downarrow$ by main induction on $r \Downarrow$, side induction on $s_i[x_i := r]\vec{S} \Downarrow$, side side induction on $s_{1-i}\vec{S} \Downarrow$ and a third side induction on the length of $\vec{S}$. We have to show $t \Downarrow$ for every reduct t of $(\mathrm{inj}_i r)(x_0.s_0, x_1.s_1)\vec{S}$. The most interesting case is when $\vec{S} \equiv T\vec{T}$ and $(x_0.s_0, x_1.s_1)$ is permuted with T, i.e., $t = (\mathrm{inj}_i r)(x_0.s_0T, x_1.s_1T)\vec{T}$. This is an application of the third side induction hypothesis since $(s_iT)[x_i := r]\vec{T} = s_i[x_i := r]\vec{S} \Downarrow, s_{1-i}T\vec{T} \Downarrow$ and $r \Downarrow$.[25]

"$\supseteq$" (completeness): Use nested course-of-generation inductions on $r \Downarrow$ and r. Actually, we may also apply the general proposition of 3.4. To this end define $\rhd$ as follows.

$$\begin{array}{rr} \lambda xr \rhd r, & \mathrm{inj}_i r \rhd r, \\ x\vec{r}\,[R] \rhd r_k, & x\vec{r}\,(x_0.s_0, x_1.s_1) \rhd s_i, \\ (\lambda xr)s\vec{S} \rhd s, & (\mathrm{inj}_i r)(x_0.s_0, x_1.s_1)\vec{S} \rhd s_{1-i}\vec{S} \text{ and } r. \end{array}$$

Clearly $\mathrm{WF}_\rhd = \Lambda^{\to +}$, since the size of the terms decreases in each clause. Commutation of $\rhd$ with $\to^+$ is shown by simple case analysis where the crucial case necessitates $\to^+$ instead of $\to$: $(\mathrm{inj}_i r)(x_0.s_0, x_1.s_1)\vec{S} \rhd s_{1-i}\vec{S} \to^+ t$ implies

$$(\mathrm{inj}_i r)(x_0.s_0, x_1.s_1)\vec{S} \to^* (\mathrm{inj}_i r)(x_0.s_0\vec{S}, x_1.s_1\vec{S}) \to^+ \left\{ \begin{array}{l} (\mathrm{inj}_0 r)(x_0.s_0\vec{S}, x_1.t) \\ (\mathrm{inj}_1 r)(x_0.t, x_1.s_1\vec{S}) \end{array} \right\} \rhd t.$$

Proposition 3.4 yields $\mathrm{WF}_{\to^+} \subseteq \mathrm{WF}_{\rhd \cup \to^+}$. Using properties of WF we get

$$\mathrm{WF}_\to = \mathrm{WF}_{\to^+} \subseteq \mathrm{WF}_{\rhd\cup\to^+} \subseteq \mathrm{WF}_{\rhd\cup\to} \subseteq \mathrm{SN}.$$

□

Lemma. *$r(x_0.s_0S, x_1.s_1S)\vec{S} \in SN \Longrightarrow r(x_0.s_0, x_1.s_1)S\vec{S} \in SN$.*

[25] This case requires $s_{1-i}\vec{S} \Downarrow$ as second premise, justifying the somewhat peculiar fact that (β_+) goes back to $s_{1-i}\vec{S}$ instead of the subterm s_{1-i} of $(\mathrm{inj}_i r)(x_0.s_0, x_1.s_1)\vec{S}$. In an untyped calculus the term $(\mathrm{inj}_0 x)(x_0.y, x_1.\lambda z.zz)(\lambda z.zz)$ would be a counter-example to $\mathrm{SN} \subseteq \mathrm{WF}_\to$ with the wrong formulation: The permutative reduct has $\mho$ as a subterm.

Proof. By course-of-generation-induction on $r(x_0.s_0S, x_1.s_1S)\vec{S} \in \mathrm{SN}$. Distinguish cases according to the shape of r. Most cases are a trivial application of the induction hypothesis. Interesting are

Case $x\vec{r}(y_0.t_0, y_1.t_1)$. Assume $x\vec{r}(y_0.t_0, y_1.t_1)(x_0.s_0S, x_1.s_1S)\vec{S} \in \mathrm{SN}$. This has been concluded by repeated application of (Var_π) from

$$x\vec{r}(y_0.t_0(x_0.s_0S, x_1.s_1S)\vec{S}, y_1.t_1(x_0.s_0S, x_1.s_1S)\vec{S}) \in \mathrm{SN},$$

i.e., we have got $\vec{r} \in \mathrm{SN}$ and both $t_i(x_0.s_0S, x_1.s_1S)\vec{S} \in \mathrm{SN}$ before. By induction hypothesis we get $t_i(x_0.s_0, x_1.s_1)S\vec{S} \in \mathrm{SN}$. Hence

$$x\vec{r}(y_0.t_0(x_0.s_0, x_1.s_1)S\vec{S}, y_1.t_1(x_0.s_0, x_1.s_1)S\vec{S}) \in \mathrm{SN}.$$

Using the rule (Var_π) repeatedly we obtain

$$x\vec{r}(y_0.t_0, y_1.t_1)(x_0.s_0, x_1.s_1)S\vec{S} \in \mathrm{SN}.$$

Case $\mathrm{inj}_i r$. Assume $(\mathrm{inj}_i r)(x_0.s_0S, x_1.s_1S)\vec{S} \in \mathrm{SN}$. This has been derived from $(s_iS)[x_i := r]\vec{S} = s_i[x_i := r]S\vec{S} \in \mathrm{SN}$ and $r, s_{1-i}S\vec{S} \in \mathrm{SN}$. Hence $(\mathrm{inj}_i r)$ $(x_0.s_0, x_1.s_1)S\vec{S} \in \mathrm{SN}$. □

5.7. Closure properties

Lemma. *For all types ρ, for all $r \in SN$,*

(i) if $s^\rho \in SN$ then $rs \in SN$,
(ii) if $r : \rho \equiv \rho_0 + \rho_1$ and $s_0, s_1 \in SN$ then $r(x_0.s_0, x_1.s_1) \in SN$,
(iii) if $s^\rho \in SN$ then $r_x[s] \in SN$.

Proof. By simultaneous induction on ρ, side induction on $r \in \mathrm{SN}$. Distinguish cases according to $r \in \mathrm{SN}$. We always first prove (i) and (ii) in parallel and later infer (iii), possibly with the help of (ii).

$x\vec{r}\,[R]$. Let $S \in \mathrm{SN}$. If $[R]$ is void then $x\vec{r}S \in \mathrm{SN}$ simply follows from $\vec{r}, S \in \mathrm{SN}$. Otherwise we may assume that R is a critical elimination of the form $(x_0.s_0, x_1.s_1)$. For $x\vec{r}(x_0.s_0, x_1.s_1)S \in \mathrm{SN}$ we need $x\vec{r}(x_0.s_0S, x_1.s_1S) \in \mathrm{SN}$ and this requires $s_0S, s_1S \in \mathrm{SN}$. But the latter follows from the side induction hypothesis for s_0 and s_1 which in case that S is critical applies since they have the same type as $x\vec{r}(x_0.s_0, x_1.s_1)$. In case (i) we do not need this last fact. Note that in order to use the side induction hypothesis (ii) we always have to check that the previously generated term in SN has the same (sum) type ρ.

$x\vec{r}(x_0.s_0, x_1.s_1)T\vec{T} \in \mathrm{SN}$ has been derived from $x\vec{r}(x_0.s_0T, x_1.s_1T)\vec{T} \in \mathrm{SN}$. The side induction hypothesis for the latter yields $x\vec{r}(x_0.s_0T, x_1.s_1T)\vec{T}S \in \mathrm{SN}$, so we obtain $x\vec{r}(x_0.s_0, x_1.s_1)T\vec{T}S \in \mathrm{SN}$ at once.

$\lambda xr \in \mathrm{SN}$ has been derived from $r \in \mathrm{SN}$. Since λxr necessarily has a function type we only have to deal with case (i) and may apply the side induction hypothesis for (iii) in order to conclude $r_x[s] \in \mathrm{SN}$. Hence $(\lambda xr)s \in \mathrm{SN}$.

$(\lambda xr)t\vec{T}$. If $S \in \mathrm{SN}$ then by the respective side induction hypothesis $r_x[t]\vec{T}S \in \mathrm{SN}$, hence $(\lambda xr)t\vec{T}S \in \mathrm{SN}$ by ($\beta_\to$).

$\mathrm{inj}_i r \in \mathrm{SN}$ is derived from $r \in \mathrm{SN}$. Since $\mathrm{inj}_i r$ has a sum type only case (ii) is possible. The assertion $(\mathrm{inj}_i r)(x_0.s_0, x_1.s_1) \in \mathrm{SN}$ can be obtained from $s_i[x_i := r] \in \mathrm{SN}$ using the (β_+)-rule, and that follows from the main induction hypothesis (iii) since r has a strictly smaller type than $\mathrm{inj}_i r$.

$(\mathrm{inj}_i r)(x_0.s_0, x_1.s_1)\vec{S}$. Similar to the $(\beta_\rightarrow)$-case: $(\mathrm{inj}_i r)(x_0.s_0, x_1.s_1)\vec{S} \in \mathrm{SN}$ has been inferred from $s_i[x_i := r]\vec{S} \in \mathrm{SN}$, $s_{1-i}\vec{S} \in \mathrm{SN}$ and $r \in \mathrm{SN}$. Since the first two terms also have type ρ the side induction hypothesis may be applied and yields

$$s_i[x_i := r]\vec{S}S \in \mathrm{SN} \qquad \text{and} \qquad s_{1-i}\vec{S}S \in \mathrm{SN},$$

so one application of the (β_+)-rule proves the assertion

$$(\mathrm{inj}_i r)(x_0.s_0, x_1.s_1)\vec{S}S \in \mathrm{SN}.$$

In the proof of (iii) the complicated cases are the variable eliminations (all other cases are covered by the side induction hypothesis).

Case $y\vec{r}\,[R]$. The side induction hypothesis supplies us with $\vec{r}_x[s] \in \mathrm{SN}$ and $R_x[s] \in \mathrm{SN}$.

- If $x \not\equiv y$ this suffices to conclude $(y\vec{r}\,[R])_x[s] \in \mathrm{SN}$.
- If $x \equiv y$ and the elimination is empty the claim follows at once from $s \in \mathrm{SN}$.
- If not we successively apply s to the substituted eliminations $\vec{r}_x[s]$, using the induction hypothesis for (i) repeatedly at subtypes of ρ, arriving at $s\vec{r}_x[s] \in \mathrm{SN}$. If $[R]$ is not void we may assume that R is critical. If, moreover, $\vec{r}$ is empty we apply (ii) at type ρ (which already has been proved) in order to conclude that $sR_x[s] \in \mathrm{SN}$. Otherwise $s\vec{r}_x[s]$ has a subtype of ρ and we apply the induction hypothesis for (ii), obtaining $s\vec{r}_x[s]R_x[s] \in \mathrm{SN}$.
  For this to work it is vital that at most the last elimination is critical, because in contrast to $\rightarrow$-eliminations the types are not in general lowered by $+$-eliminations.

Case $y\vec{r}(x_0.s_0, x_1.s_1)S\vec{S}$. By induction hypothesis $(y\vec{r}(x_0.s_0S, x_1.s_1S)\vec{S})_x[s] \in \mathrm{SN}$. In the more difficult case of $y \equiv x$ this term has the form

$$s\vec{r}'(x_0.s_0'S', x_1.s_1'S')\vec{S}'.$$

Using the previous lemma we obtain $s\vec{r}'(x_0.s_0', x_1.s_1')S'\vec{S}' \in \mathrm{SN}$. □

Corollary. $r \Downarrow$.

5.8. On η

Let now $\rightarrow_\eta$ be generated from the eta conversion rule for function types $\lambda x.rx \rightarrow r$ (if x is not free in r) as before and the eta conversion rule for sum types $r(x_0.\mathrm{inj}_0 x_0, x_1.\mathrm{inj}_1 x_1) \rightarrow r$ and write $\rightarrow_{\beta\pi\eta}$ for $\rightarrow \cup \rightarrow_\eta$. As in 3.5, $\mathrm{WF}_{\rightarrow_{\beta\pi\eta}} \subseteq \mathrm{SN}$ by antitonicity of $\lambda\succ.\mathrm{WF}_\succ$. The converse inclusion $\mathrm{SN} \subseteq \mathrm{WF}_{\rightarrow_{\beta\pi\eta}}$ is shown by a slight extension of the soundness proof. We only discuss the interesting new cases:

(Var$_\pi$). If $x\vec{r}(x_0.s_0, x_1.s_1)S\vec{S} \to_\eta x\vec{r}S\vec{S} =: t$ due to $s_i = \mathrm{inj}_i x_i$, then S has the form $S \equiv (x_0.t_0, x_1.t_1)$[26] and

$$x\vec{r}(x_0.s_0S, x_1.s_1S)\vec{S} \equiv x\vec{r}(x_0.(\mathrm{inj}_0x_0)(x_0.t_0, x_1.t_1), x_1.(\mathrm{inj}_1x_1)(x_0.t_0, x_1.t_1))\vec{S}$$

reduces with two β_+-steps to t, hence $t \Downarrow$ follows from the assumption alone.

If $x\vec{r}(x_0.s_0, x_1.s_1)S\vec{S} \to_\eta x\vec{r}(x_0.s_0, x_1.s_1)\vec{S} =: t$ due to $S \equiv (y_0.\mathrm{inj}_0y_0, y_1.\mathrm{inj}_1y_1)$, then $x\vec{r}(x_0.s_0S, x_1.s_1S)\vec{S}$ reduces to t in two η-steps, and therefore we again do not need the induction hypothesis.

(β_+). If $(\mathrm{inj}_ir)(x_0.s_0, x_1.s_1)\vec{S} \to_\eta (\mathrm{inj}_ir)\vec{S}$ due to $s_j = \mathrm{inj}_jx_j$ for $j \in \{0, 1\}$, then $(\mathrm{inj}_ir)\vec{S} = s_i[x_i := r]\vec{S} \Downarrow$ by assumption.

We already know that $\Lambda^{\to +} \subseteq \mathrm{SN}$, hence also $\to_{\beta\pi\eta}$ is strongly normalizing.[27]

6. Extension to generalized applications

Generalized applications allow to further simplify the grammar of normal forms in λ-calculus. The term system presented here is inspired by von Plato's generalized natural deduction trees [vP01].

Its typed version contains $\to$ as the only type constructor, although the proofs directly carry over to a system with products (with a generalization of projections) and sums (with the usual elimination rule, treated above). In fact, the proofs in this section for $\to$ are guided by those of the preceding section for $+$.

6.1. (Raw) terms

$$\Lambda\mathrm{J} \ni r, s, t ::= x \mid \lambda yr \mid s(t, y.r).$$

The variable y gets bound in r in these terms. By setting $rs := r(s, y.y)$, Λ embeds into ΛJ.

6.2. Inductive characterization and normal forms

We use capital letters R, S, T for eliminations, i. e., for syntactic objects of the form $(s, z.t)$. We will write $R_x[r]$ for the capture-free substitution $(s_x[r], z.t_x[r])$. The set of terms is inductively characterized by the following grammar —

$$\Lambda\mathrm{J} \ni r ::= x \mid xR \mid \lambda xr \mid xRS\vec{S} \mid (\lambda xr)S\vec{S},$$

where the last two forms capture non-normal terms: $xRS\vec{S}$ is a permutative redex and $(\lambda xr)S\vec{S}$ is a β-redex (see below). By excluding these last clauses we obtain the grammar of normal forms

$$\mathrm{NF} \ni r, s, t ::= x \mid x(s, z.t) \mid \lambda xr.$$

[26] Concerning the choice of variable names, see footnote 11.

[27] The double-negation-translation used in [dG99] for establishing strong normalization of $\to$ does not cover $\to_\eta$.

6.3. *Types and type assignment*

By using the types and type assignments for variables from 1.3, the typable terms and their unique type are given by the rules:

$$\frac{r : \rho \to \sigma \qquad s : \rho \qquad t : \tau}{r(s, z^{\sigma}.t) : \tau} \qquad \frac{r : \sigma \qquad x : \rho}{\lambda x r : \rho \to \sigma}$$

(The type superscript of z indicates the type assignment for z and could have been written as an extra premise.) By restricting the first rule to $r \equiv x$, we get a typing of the normal forms and arrive at a term notation for the cut-free Gentzen calculus (describing the implicational fragment of intuitionistic propositional logic), the first rule being the left $\to$-introduction rule.[28] Therefore, the typing rules are correct from a logical point of view.

Induction on NF verifies the subformula property of normal terms.[29]

6.4. *Reduction*

The conversions to be dealt with are β-conversion

$$(\lambda x r)(s, z.t) \to_{\beta} t[z := r_x[s]]$$

and permutative conversion

$$r(s, z.t)S \to_{\pi} r(s, z.tS).^{30}$$

Their *compatible closure* $\to$ is given by

$$r \to r' \Longrightarrow rS \to r'S, s(r, z.t) \to s(r', z.t), s(t, z.r) \to s(t, z.r'), \lambda x r \to \lambda x r'.$$

Clearly, reduction preserves typability and types, i. e., we have subject reduction for $\to$.

[28] An alternative notation for $x(s, z.t)$ proposed in [Sch99] is $t_z\{x, s\}$ which is motivated by the following type-preserving translation into the normal forms of simply-typed λ-calculus: $F(x) := x$, $F(\lambda x r) := \lambda x F(r)$, $F(x(s, z.t)) := F(t)[z := xF(s)]$. Its natural extension $F(r(s, z.t)) := F(t)[z := F(r)F(s)]$ translates ΛJ into Λ, but does not simulate reduction, so it cannot serve to infer strong normalization from $\Lambda^{\to}$. Neither can the variant with $F(r(s, z.t)) := (\lambda z F(t))(F(r)F(s))$.

[29] $xRS\vec{S}$ need not have the subformula property: $x_1^{\iota\to\iota}(y_1^{\iota}, z^{\iota}.\lambda x^{\rho\to\rho} z)(\lambda y^{\rho} y, z^{\iota}.z) : \iota$ has the subterm $\lambda y^{\rho} y$ of type $\rho \to \rho$ with arbitrary type ρ but only the free variables x_1 and y_1 of types $\iota \to \iota$ and ι.

[30] z shall not be free in S, compare footnote 23.

6.5. Definition

The set SN is generated inductively by the following rules.

$$\frac{}{x \in \mathrm{SN}}(\mathrm{Var}_0) \qquad \frac{r \in \mathrm{SN}}{\lambda x r \in \mathrm{SN}}(\lambda)$$

$$\frac{s, t \in \mathrm{SN}}{x(s, z.t) \in \mathrm{SN}}(\mathrm{Var}) \qquad \frac{x(s, z.tS)\vec{S} \in \mathrm{SN}}{x(s, z.t)S\vec{S} \in \mathrm{SN}}(\mathrm{Var}_\pi)^{31}$$

$$\frac{t[z := r_x[s]]\vec{S} \in \mathrm{SN} \qquad r, s \in \mathrm{SN}}{(\lambda x r)(s, z.t)\vec{S} \in \mathrm{SN}}(\beta)$$

Lemma. *$SN = WF_{\rightarrow}$.*

Proof. "$\subseteq$" (soundness): Once again we show that $\mathrm{WF}_{\rightarrow} = \{r \mid r \Downarrow\}$ has the closure properties which define SN. (Var_0) is trivial, for (Var) and (λ) we essentially take the proofs of section 3. (Var_π) works in nearly the same way as in the preceding section. The proof for (β) is a combination of (β) in section 3 with (β_+) in section 5. The last two properties are shown in detail for reference purposes.

(Var_π). Show $x(s, z.tS)\vec{S} \Downarrow \Longrightarrow x(s, z.t)S\vec{S} \Downarrow$ by induction on $x(s, z.tS)\vec{S} \in \mathrm{WF}_{\rightarrow^+}$. We have to show $r \Downarrow$ for every reduct r of $x(s, z.t)S\vec{S}$. The only interesting case arises with $\vec{S} \equiv T\vec{T}$ and a permutation of $S \equiv (s', z'.t')$ with T leading to $r = x(s, z.t)(s', z'.t'T)\vec{T}$. $r \Downarrow$ follows by induction hypothesis as $x(s, z.tS)\vec{S} \equiv x(s, z.t(s', z'.t'))T\vec{T}$ reduces with two permutative reduction steps (hence once $\rightarrow^+$) to $x(s, z.t(s', z'.t'T))\vec{T}$.

(β). Show that $t[z := r_x[s]]\vec{S} \Downarrow$, $r \Downarrow$ and $s \Downarrow$ imply $(\lambda x r)(s, z.t)\vec{S} \Downarrow$ by main induction on $s \Downarrow$, side induction on $r \Downarrow$, side side induction on $t[z := r_x[s]]\vec{S} \Downarrow$ and a third side induction on the length of $\vec{S}$. We have to show $r' \Downarrow$ for every reduct r' of $(\lambda x r)(s, z.t)\vec{S}$. In case of the outer β-reduction the assumption is used; reduction in s needs the main induction hypothesis, reduction in r the side induction hypothesis. Reductions in t and in $\vec{S}$ as well as permutations between $\vec{S}$ are covered by the side side induction hypothesis. The remaining case is when $\vec{S} \equiv T\vec{T}$ and $(s, z.t)$ is permuted with T, i.e., $r' = (\lambda x r)(s, z.tT)\vec{T}$. This is an application of the third side induction hypothesis since $(tT)[z := r_x[s]]\vec{T} = t[z := r_x[s]]\vec{S}$.

"$\supseteq$" (completeness): Once again use nested course-of-generation inductions on $r \Downarrow$ and r or apply the general proposition of 3.4 with $\rhd$ defined as follows:

$$x(s, z.t) \rhd s \text{ and } t, \qquad \lambda x r \rhd r, \qquad \text{and} \qquad (\lambda x r)(s, z.t)\vec{S} \rhd r \text{ and } s. \qquad \square$$

Lemma. *$r(s, z.tS)\vec{S} \in SN \Longrightarrow r(s, z.t)S\vec{S} \in SN$.*

Proof. By course-of-generation-induction on $r(s, z.tS)\vec{S} \in \mathrm{SN}$. Distinguish cases according to the shape of r. The last two cases of the grammar in 6.2 are a simple application of the induction hypothesis, the case of a variable is rule (Var_π). Interesting are

[31] This clause is subject to the same proviso as the rule of permutative conversion.

Case $x(s', z'.t')$. Assume $x(s', z'.t')(s, z.tS)\vec{S} \in \mathrm{SN}$. This has been concluded by repeated application of (Var_π) from $x(s', z'.t'(s, z.tS)\vec{S}) \in \mathrm{SN}$, i. e., we have got $s' \in \mathrm{SN}$ and $t'(s, z.tS)\vec{S} \in \mathrm{SN}$ before. By induction hypothesis we get $t'(s, z.t)S\vec{S} \in \mathrm{SN}$. Hence $x(s', z'.t'(s, z.t)S\vec{S}) \in \mathrm{SN}$. Using the rule ($\mathrm{Var}_\pi$) repeatedly we obtain $x(s', z'.t')(s, z.t)S\vec{S} \in \mathrm{SN}$.

Case λxr. Assume $(\lambda xr)(s, z.tS)\vec{S} \in \mathrm{SN}$. This has been derived from $r \in \mathrm{SN}$, $s \in \mathrm{SN}$ and $(tS)[z := r_x[s]]\vec{S} = t[z := r_x[s]]S\vec{S} \in \mathrm{SN}$. Hence $(\lambda xr)(s, z.t)S$ $\vec{S} \in \mathrm{SN}$. □

6.6. *Strong normalization*

For the rest of this section we assume that all displayed terms are typable.

Lemma. *For all types ρ, for all $r \in SN$,*

(i) if $r : \rho \equiv \rho_0 \to \rho_1$ and $s, t \in SN$ then $r(s, z.t) \in SN$,
(ii) if $s^\rho \in SN$ then $r_x[s] \in SN$.

Proof. By simultaneous induction on ρ, side induction on $r \in \mathrm{SN}$. Distinguish cases according to $r \in \mathrm{SN}$. One always first has to prove (i) and later infers (ii), possibly with the help of (i). (i). Case (Var_0) only needs (Var); (Var), (Var_π) and (β) are covered by the side induction hypothesis (check that the type of the main induction is not affected) and the rules (Var_π)&(Var_0), (Var_π) and (β), respectively. (ii). Case (Var_0) needs either the assumption (if $r \equiv x$) or rule (Var_0) again; (Var_π), (λ) and (β) follow from the side induction hypothesis and the previous lemma, rules (λ) and (β), respectively. Therefore, only (i)(λ) and (ii)(Var) are studied in detail. The reader is invited to contrast them with case λxr and the third subcase of $y\vec{r}\ [R]$ in the proof of 5.7.

(i)(λ). Let $\lambda xr : \rho_0 \to \rho_1$ be in SN due to $r^{\rho_1} \in \mathrm{SN}$. Show that $(\lambda xr)(s^{\rho_0}, z.t) \in \mathrm{SN}$. By rule ($\beta$) it suffices to show $r, s \in \mathrm{SN}$ and $t[z := r_x[s]] \in \mathrm{SN}$. By induction hypothesis (ii) for type ρ_0, $r_x[s] \in \mathrm{SN}$. $r_x[s] : \rho_1$, hence by induction hypothesis (ii) for type ρ_1, $t[z := r_x[s]] \in \mathrm{SN}$.

(ii)(Var). Let $y(s', z'.t')$ be in SN due to $s', t' \in \mathrm{SN}$. Show that $(y(s', z'.t'))_x[s] \in \mathrm{SN}$. By side induction hypothesis (ii), $s'_x[s], t'_x[s] \in \mathrm{SN}$. If $y \not\equiv x$, we are done by rule (Var). Otherwise, we have to show $s(s'_x[s], z'.t'_x[s]) \in \mathrm{SN}$. This is an instance of (i) for the same type ρ which is proved before (ii) and hence already available. □

Corollary. $r \Downarrow$.

6.7. *On κ*

Following the idea of "immediate simplifications" in [Pra71, p. 254] which allow to reduce $+$-eliminations $r(x_0.s_0, x_1.s_1)$ to the term s_i if x_i does not occur free in s_i (for $i \in \{0, 1\}$), define the new conversion rule $r(s, z.t) \to t$, if z is not free in t and write $\to_\kappa$ for its compatible closure and $\to_{\beta\pi\kappa}$ for $\to \cup \to_\kappa$. Then also $\to_{\beta\pi\kappa}$ is strongly normalizing because $\mathrm{WF}_{\beta\pi\kappa}$ has the defining closure properties of SN: (Var$_0$) and (λ) are trivial as before; (Var) may in addition have an outer κ-conversion which is trivial to handle, though. The proofs for (Var$_\pi$) and (β) in 6.5 have to be extended by some cases:

(Var$_\pi$). If z is not free in t and $r = tS\vec{S}$, then also $x(s, z.tS)\vec{S} \rightarrow_\kappa r$. If $S \equiv (s', z'.t')$ with z' not free in t' and $r = t'\vec{S}$, then $x(s, z.tS)\vec{S} \rightarrow_\kappa x(s, z.t')\vec{S} \rightarrow_\kappa r$ since we may assume that z is not free in t'. Finally, if $\vec{S} \equiv \vec{R}(s', z'.t')\vec{T}$ with z' not free in t' and $r = t'\vec{T}$, then $x(s, z.tS)\vec{S} \rightarrow_\kappa r$. Note that the induction hypothesis is never needed in these new cases.

(β). If z is not free in t and $r' = t\vec{S}$, then this is also a β-reduction (an analogous situation to that of η in 3.5). If $\vec{S} \equiv \vec{R}(s', z'.t')\vec{T}$ with z' not free in t' and $r = t'\vec{T}$, then also $t[z := r_x[s]]\vec{S} \rightarrow_\kappa r$. □

7. Gödel's T

Compared with a combinatory version [Tai67, San67, Wei97], primitive recursion in finite types over the λ-calculus presents additional difficulties in normalization proofs: β-reduction might substitute variables in the recursion arguments, giving rise to new recursion steps. This mixing of β- and recursion steps can be avoided by adapting a special reduction strategy [Dil68, How80] or restricting recursion arguments to numerals only.[32] In this section we show strong normalization for the general form of a λ-calculus enriched by number and recursion constructors as well as recursion reductions. In order to keep the presentation short we directly proceed to the typed calculus although an untyped version of Gödel's T can be described (Example 3.3.5 in [Joa01]).

7.1. Terms and Types

We formulate T by adding a basic type N for the natural numbers, a constant $0 : \mathsf{N}$ and a unary function symbol $\mathcal{S} : \mathsf{N} \rightarrow \mathsf{N}$ for the successor. Higher-order recursion is implemented by the elimination rule

$$\frac{r : \mathsf{N} \qquad s : \rho \qquad t : \mathsf{N} \rightarrow \rho \rightarrow \rho}{r(s, t) : \rho}$$

We identify natural numbers n with the respective numerals $\underbrace{\mathcal{S} \ldots \mathcal{S}}_{n \text{ times}} 0$.

7.2. Inductive characterization and normal forms

Once more we extend the notion of elimination to capture N-recursion:

$$R, S ::= r \mid (r, s).$$

With this notation we can characterize terms by

$$\mathsf{T} \ni r, s ::= x\vec{R} \mid \lambda xr \mid 0 \mid \mathcal{S}r \mid 0(r, s)\vec{S} \mid (\mathcal{S}r)(s, t)\vec{S} \mid (\lambda xr)s\vec{S}$$

and normal forms by

$$\mathrm{NF} \ni r, s ::= x\vec{R} \mid \lambda xr \mid 0 \mid \mathcal{S}r \quad \text{with } R ::= r \mid (r, s).$$

[32] In combinatory versions of T the latter restriction is quite common [San67, Sch77].

7.3. Reduction

Additional conversion rules are

$$0(r, s) \to r \qquad (\mathcal{S}r)(s, t) \to tr(r(s, t)).$$

The *compatible closure* $\to$ is defined as expected:

$$\begin{aligned} r \to r' \Longrightarrow t(r, s) \to t(r', s), t(s, r) \to t(s, r'), \\ rR \to r'R, \mathcal{S}r \to \mathcal{S}r', \lambda xr \to \lambda xr'. \end{aligned}$$

7.4. Definition

$(r, s) \in \mathrm{SN}$ is an abbreviation for $r \in \mathrm{SN}$ *and* $s \in \mathrm{SN}$.

$$\frac{}{0 \in \mathrm{SN}}(0) \qquad \frac{r \in \mathrm{SN}}{\mathcal{S}r \in \mathrm{SN}}(\mathcal{S}) \qquad \frac{r \in \mathrm{SN}}{\lambda xr \in \mathrm{SN}}(\lambda)$$

and for recursion

$$\frac{r\vec{S} \in \mathrm{SN} \qquad s \in \mathrm{SN}}{0(r, s)\vec{S} \in \mathrm{SN}}(R_0) \qquad \frac{tr(r(s, t))\vec{S} \in \mathrm{SN}}{(\mathcal{S}r)(s, t)\vec{S} \in \mathrm{SN}}(R_{\mathcal{S}})$$

The rules (Var) and (β) now allow for N–eliminations.

$$\frac{\vec{R} \in \mathrm{SN}}{x\vec{R} \in \mathrm{SN}}(\mathrm{Var}) \qquad \frac{r_x[s]\vec{S} \in \mathrm{SN} \qquad s \in \mathrm{SN}}{(\lambda xr)s\vec{S} \in \mathrm{SN}}(\beta)$$

Lemma. *$SN = WF_{\to}$.*

Proof. Parallel to the proofs in section 3, except for the rule ($R_{\mathcal{S}}$) in the proof of "$\subseteq$" where induction on $tr(r(s, t))\vec{S} \in \mathrm{WF}_{\to^+}$ is needed, because t and r occur twice in this term. □

Corollary. *SN is closed under the subterm relation.*

7.5. The set Ω

We cannot (hope to) extend the simple proof of closure under substitution for SN for typed terms due to the variable rule. Therefore we investigate a subset of SN, termed Ω, for which we can show this property.

Definition. *The set Ω is generated by the same rules as SN, where (Var) and ($R_{\mathcal{S}}$) are restricted*

$$\frac{\vec{r} \in \Omega}{x\vec{r} \in \Omega}(\mathit{Var}) \qquad \frac{tn(n(s, t))\vec{T} \in \Omega}{(\mathcal{S}n)(s, t)\vec{T} \in \Omega}(R_{\mathcal{S}})$$

and the following infinitary ω-rule is added.

$$\frac{r \in \Omega \qquad \cdots\ n(s, t)\vec{T} \in \Omega\ \cdots_{n\in\mathbb{N}}}{r(s, t)\vec{T} \in \Omega}(\omega)$$

Remark. N-eliminations of a variable can only enter Ω through the ω-rule.[33]

[33] Thus in the untyped calculus the term $x(\lambda z.zz, y)(\lambda z.zz)$ would be in $\mathrm{SN} \setminus \Omega$.

Lemma. *$r \in \Omega, s^\rho \in \Omega \implies$ (i) $rs \in \Omega$, (ii) $r_x[s] \in \Omega$.*

Proof. We use simultaneous induction on ρ and side induction on the generation of $r \in \Omega$. We only cover the new cases. (0) is trivial. $(\mathcal{S})$, (R_0) and $(R_{\mathcal{S}})$ follow directly from the side induction hypotheses. So consider the rule (ω), where $r(t_0, t_1)\vec{T} \in \Omega$ is inferred from $r \in \Omega$ and infinitely many assumptions $r_n := n(t_0, t_1)\vec{T} \in \Omega$.

(i) The side induction hypothesis shows $r_n s \in \Omega$ for each $n \in \mathbb{N}$. Hence we may apply rule (ω) in order to obtain $r(t_0, t_1)\vec{T}s \in \mathrm{SN}$.
(ii) The side induction hypothesis yields $r_x[s] \in \Omega$ and $\forall n.r_n[x := s] \in \Omega$. Thus one application of the rule (ω) proves $(r(t_0, t_1)\vec{T})_x[s] \in \Omega$. □

Corollary. $r \in \Omega$.

Proof. By induction on r. The interesting new case is an N-elimination (where we use the ω-rule): Let $r : \mathsf{N}, s : \rho, t : \mathsf{N} \to \rho \to \rho$ be in Ω. In order to conclude $r(s, t) \in \Omega$ it suffices to show $\forall n.n(s, t) \in \Omega$. This is done by side induction on n.

Case 0. $0(s, t) \in \Omega$ is inferred from $s, t \in \Omega$ by the rule (R_0).
Case $n+1$. By the side induction hypothesis for n we have $n(s, t) \in \Omega$. Two applications of lemma (i) yield $tn(n(s, t)) \in \Omega$, to which the rule $(R_{\mathcal{S}})$ applies. □

7.6. Injecting Ω into SN

The rules of Ω are restrictions of SN–rules, except for (ω) which is a weakened version of the following rule of *value expansion*

$$\frac{r \in \mathrm{SN} \qquad |r|(s, t)\vec{S} \in \mathrm{SN}}{r(s, t)\vec{S} \in \mathrm{SN}}\,(*)$$

where $|r|$ is defined recursively for $r^{\mathsf{N}} \in \mathrm{SN}$ by

$$\begin{aligned} |x\vec{R}| &:= 0, \\ |0| &:= 0, \\ |\mathcal{S}r| &:= \mathcal{S}|r|, \\ |(\lambda x r)s\vec{S}| &:= |r_x[s]\vec{S}|, \\ |0(r, s)\vec{S}| &:= |r\vec{S}|, \\ |(\mathcal{S}r)(s, t)\vec{S}| &:= |tr(r(s, t))\vec{S}|. \end{aligned}$$

Therefore Ω can be embedded into SN extended by rule $(*)$. The following lemma shows that $(*)$ is already admissible in SN, hence $\Omega \subset \mathrm{SN}$.

Lemma. *(Value expansion). $\vec{s}^{\,\mathsf{N}} \in SN$, $r[\vec{x} := |\vec{s}\,|] \in SN \implies r[\vec{x} := \vec{s}\,] \in SN$.*

Proof. We abbreviate $r^{\vec{t}} := r_{\vec{x}}[\vec{t}\,]$ and set $\vec{n} := |\vec{s}\,|$. The assertion is proved by induction on the generation of $r^{\vec{n}} \in \mathrm{SN}$.

Case ($\mathcal{S}$). Assume $r^{\vec{n}} = \mathcal{S}t \in \mathrm{SN}$ has been concluded from $t \in \mathrm{SN}$. If $r = x_k$ then t is a numeral and $r^{\vec{s}} = x_k^{\vec{s}} = s_k$, so the claim is trivial. Otherwise $r = \mathcal{S}t_0$ with $t_0^{\vec{n}} = t$ and the induction hypothesis shows $t_0^{\vec{s}} \in \mathrm{SN}$, hence also $\mathcal{S}t_0^{\vec{s}} \in \mathrm{SN}$.

Case (0). Analogous, but simpler.

Case (Var). $r^{\vec{n}} = x\vec{R} \in \mathrm{SN}$ has been concluded from $\vec{R} \in \mathrm{SN}$. x cannot be in $\vec{x}$ and we only need to use the induction hypothesis for $\vec{R}$.

Case (R_0). Assume $r^{\vec{n}} = 0(r'_0, r'_1)\vec{R}' \in \mathrm{SN}$ has been concluded from $r'_0\vec{R}', r'_1 \in \mathrm{SN}$. Now if $r = 0(r_0, r_1)\vec{R}$ then the claim follows easily from the induction hypothesis for $r'_0\vec{R}', r'_1$. Otherwise $r = x_k(r_0, r_1)\vec{R}, n_k = 0$ and we have to show

$$s_k(r_0^{\vec{s}}, r_1^{\vec{s}})\vec{R}^{\vec{s}} \in \mathrm{SN}.$$

We derive this as an instance of

$$\forall \hat{s}^{\mathsf{N}} \in \mathrm{SN}.|\hat{s}| = 0 \Longrightarrow \hat{s}(r_0^{\vec{s}}, r_1^{\vec{s}})\vec{R}^{\vec{s}} \in \mathrm{SN},$$

which is proved by side induction on $\hat{s} \in \mathrm{SN}$. Since $\hat{s} : \mathsf{N}$ and $|\hat{s}| = 0$, the cases ($\mathcal{S}$) and (λ) cannot occur. Furthermore, all reduction steps (rules (R_0), ($R_{\mathcal{S}}$), (β)) are easily reverted, so we only consider the following two cases.

Case 0. This case is identical to the one where $r = 0(r_0, r_1)\vec{R}$.

Case $x\vec{S}$. We need $x\vec{S}(r_0^{\vec{s}}, r_1^{\vec{s}})\vec{R}^{\vec{s}} \in \mathrm{SN}$, which amounts to $r_0^{\vec{s}}, \vec{R}^{\vec{s}} \in \mathrm{SN}$ and $r_1^{\vec{s}} \in \mathrm{SN}$. The latter follows from the main induction hypothesis for $r_1^{\vec{n}} = r'_1$. The other terms are in SN since $r_0^{\vec{s}}\vec{R}^{\vec{s}} \in \mathrm{SN}$ by the main induction hypothesis (applied to $r'_0\vec{R}'$) and SN is closed under the subterm relation.

Case ($R_{\mathcal{S}}$). Suppose $r^{\vec{n}} = (\mathcal{S}r')(s', t')\vec{R}' \in \mathrm{SN}$ comes from $t'r'(r'(s', t'))\vec{R}' \in \mathrm{SN}$. Now if $r = (\mathcal{S}r'')(s, t)\vec{R}$ then the claim follows easily from the induction hypothesis. Otherwise $r = x_k(s, t)\vec{R}$, n_k is the successor of — say — m and we need to show

$$s_k(s^{\vec{s}}, t^{\vec{s}})\vec{R}^{\vec{s}} \in \mathrm{SN}.$$

Similar to the case (R_0) we show the more general statement

$$\forall \hat{s}^{\mathsf{N}}.|\hat{s}| = \mathcal{S}m \Longrightarrow \hat{s}(s^{\vec{s}}, t^{\vec{s}})\vec{R}^{\vec{s}} \in \mathrm{SN}$$

by side induction on $\hat{s} \in \mathrm{SN}$. Since $|\hat{s}| = \mathcal{S}m$, the variable cases and 0 can be excluded. All reduction cases are easily handled by the side induction hypothesis. So we are left with the case ($\mathcal{S}$) where $\hat{s} = \mathcal{S}\hat{s}'$ and $|\hat{s}'| = m$. It suffices to show

$$t^{\vec{s}}\hat{s}'(\hat{s}'(s^{\vec{s}}, t^{\vec{s}}))\vec{R}^{\vec{s}} \in \mathrm{SN}.$$

This term can be expressed as $r^*_{\vec{x},x}[\vec{s}, \hat{s}']$ with $r^* = tx(x(s, t))\vec{R}$ and a new variable x. Moreover,

$$r^*_{\vec{x},x}[\vec{n}, m] = t'r'(r'(s', t'))\vec{R}',$$

so we can use the main induction hypothesis for that term to prove the claim.

The cases (λ) and (β) are simple applications of the induction hypothesis. □

Corollary. $r \Downarrow$.

Acknowledgements. We would like to thank our mentor Helmut Schwichtenberg for numerous encouraging meetings and stimulating advice. Wilfried Buchholz provided invaluable insight into inductive definitions. Jan von Plato brought to our attention the problem of strong normalization for natural deduction with general eliminations, the solution of which now forms the core of section 6. For repeated discussions and helpful remarks we are grateful to Henk Barendregt, Ulrich Berger, René David, Healfdene Goguen, J. Roger Hindley, Tobias Nipkow[34] and Anne Troelstra.

References

[AJ01] Aehlig, K., Joachimski, F.: Operational aspects of normalization by evaluation. Submitted, 2001

[Alt93] Altenkirch, T.: A formalization of the strong normalization proof for system F in LEGO. In Bezem and Groote [BG93], pages 13–28

[Bar84] Barendregt, H.P.: *The Lambda Calculus: Its Syntax and Semantics*. North-Holland, Amsterdam, second revised edition, 1984

[Ber93] Berger, U.: Program extraction from normalization proofs. In Bezem and Groote [BG93], 91–106

[BG93] Bezem, M., Jan F. Groote, editors. *Typed Lambda Calculi and Applications*, volume 664 of *Lecture Notes in Computer Science*. Springer Verlag, 1993

[Dav95] David, R.: Une preuve simple de résultats classiques en λ calcul. *C.R. Acad. Sci. Paris*, t. 320, Série I:1401–1406, (1995)

[Dav01] David, R.: Normalization without reducibility. Annals of Pure and Applied Logic **107**(1–3), 121–130 (2001)

[dG99] de Groote, P.: On the strong normalisation of natural deduction with permutation-conversions. In Paliath Narendran and Michaël Rusinowitch, editors, *Rewriting Techniques and Applications, 10th International Conference, RTA-99, Trento, Italy, July 2-4, 1999, Proceedings*, volume 1631 of *Lecture Notes in Computer Science*, 45–59. Springer Verlag, 1999

[Dil68] Diller, J.: Zur Berechenbarkeit primitiv-rekursiver Funktionale endlicher Typen. In Kurt Schütte, editor, *Contributions to Mathematical Logic*, 109–120. North–Holland, Amsterdam, 1968

[Gan80] Gandy, R.O.: An early proof of normalization. In J.P. Seldin and J.R. Hindley, editors, *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, 453–455. Academic Press, 1980

[GLT89] Girard, J.-Y., Lafont, Y., Taylor, P.: *Proofs and Types*, volume 7 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1989

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

[Gog94] Goguen, H.: *A Typed Operational Semantics for Type Theory*. PhD thesis, University of Edinburgh, August 1994. Available as LFCS Report ECS-LFCS-94-304

[Gog95] Goguen, H.: Typed operational semantics. In Mariangiola Dezani-Ciancaglini and Gordon Plotkin, editors, *Typed Lambda Calculi and Applications*, volume 902 of *Lecture Notes in Computer Science*, 186–200. Springer Verlag, 1995

[Gog99] Goguen, H.: Soundness of the logical framework for its typed operational semantics. In Jean-Yves Girard, editor, *Typed Lambda Calculi and Applications, 4th International Conference, TLCA'99, L'Aquila, Italy, April 7-9, 1999,*

[34] Nipkow verified lemmata 3.3, 3.4 and 4 with the theorem prover Isabelle and gave worthwhile hints for improvement.

Proceedings, volume 1581 of *Lecture Notes in Computer Science*, 177–197. Springer Verlag, 1999

[Hin97] Hindley, J.R.: *Basic Simple Type Theory*, volume 42 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1997

[How80] Howard, W.A.: Ordinal analysis of terms of finite type. *The Journal of Symbolic Logic*, **45**(3), 493–504 (1980)

[Joa01] Joachimski, F.: *Reduction Properties of* Π*IE-Systems*. PhD thesis, LMU München, 2001

[Lei75] Leivant, D.: Strong normalization for arithmetic (variations on a theme of Prawitz). In J. Diller and G.H. Müller, editors, *Proof Theory Symposion Kiel 1974*, volume 500 of *Lecture Notes in Mathematics*, 182–197. Springer Verlag, 1975

[Mat98] Matthes, R.: *Extensions of System F by Iteration and Primitive Recursion on Monotone Inductive Types*. Dissertation (PhD thesis), Universität München, 1998

[Mat00] Matthes, R.: Characterizing strongly normalizing terms for a lambda calculus with generalized applications via intersection types. In José D. P. Rolim, Andrei Z. Broder, Andrea Corradini, Roberto Gorrieri, Reiko Heckel, Juraj Hromkovic, Ugo Vaccaro, and Joe B. Wells, editors, *ICALP Workshops 2000, Proceedings of the Satellite Workshops of the 27th International Colloquium on Automata, Languages, and Programming, Geneva, Switzerland*, volume 8 of *Proceedings in Informatics*, 339–353. Carleton Scientific, 2000

[Pra71] Prawitz, D.: Ideas and results in proof theory. In Jens E. Fenstad, editor, *Proceedings of the Second Scandianvian Logic Symposium*, 235–307. North–Holland, Amsterdam, 1971

[San67] Sanchis, L.E.: Functionals defined by recursion. *Notre Dame Journal of Formal Logic*, VIII(3), 161–174 (1967)

[Sch77] Schütte, K.: *Proof Theory*. Springer-Verlag, Berlin, 1977

[Sch93] Schwichtenberg, H. Proofs as programs. In P. Aczel, H. Simmons, and S.S. Wainer, editors, *Proof Theory. A selection of papers from the Leeds Proof Theory Programme 1990*, 81–113. Cambridge University Press, 1993

[Sch99] Schwichtenberg, H.: Termination of permutative conversions in intuitionistic Gentzen calculi. Theoretical Computer Science **212**(1–2), 247–260 (1999)

[Tai67] Tait, W.W.: Intensional interpretations of functionals of finite type I. The Journal of Symbolic Logic **32**(2), 198–212 (1967)

[Tai75] Tait, W.W.: A realizability interpretation of the theory of species. In R. Parikh, editor, *Logic Colloquium Boston 1971/72*, volume 453 of *Lecture Notes in Mathematics*, 240–251. Springer Verlag, 1975

[Tro73] Troelstra A.S., editor: *Metamathematical Investigation of Intuitionistic Arithmetic and Analysis*, volume 344 of *Lecture Notes in Mathematics*. Springer Verlag, Berlin, Heidelberg, New York, 1973

[vdP96] van de Pol, J.: Termination of higher-order rewrite systems. Quaestiones Infinitae XVI, Department of Philosophy, Utrecht University, 1996. Proefschrift (PhD thesis)

[vP01] von Plato, J.: Natural deduction with general elimination rules. Annals of Mathematical Logic **40**(7), 541–567 (2001)

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

[vRS95] van Raamsdonk, F., Severi, P.: On normalisation. Technical Report CS-R9545, Centrum voor Wiskunde en Informatica (CWI), Amsterdam, June 1995. Forms a part of [vR96] [vRSSX99] van Raamsdonk, F., Severi, P., Sørensen, M.H.B., Xi, H. Perpetual reductions in λ-calculus. Information and Computation **149**(2), 173–225 (1999)

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