Characterizing Strongly Normalizing Terms of a λ-Calculus with Generalized Applications via Intersection Types

R. MATTHES
Ludwig-Maximilians-Universität München, Germany

Abstract

An intersection type assignment system for the extension ΛJ of the untyped λ-calculus, introduced by Joachimski and Matthes, is given and proven to characterize the strongly normalizing terms of ΛJ. Since ΛJ's generalized applications naturally allow permutative/commuting conversions, this is the first analysis of a term rewrite system with permutative conversions by help of intersection types. Two proofs are given for the fact that the typable terms are strongly normalizing: One by the computability predicates method à la Tait and one showing directly that strongly normalizing typable terms are closed under (generalized) application and substitution. It is also shown that a straightforward extension of the type assignment for λ-calculus fails to capture the strongly normalizing terms.

Keywords

Intersection Types, Strong Normalization, Permutative Conversions, Saturated Sets.

1 Introduction

In [5] an extension ΛJ of λ-calculus with generalized applications inspired by von Plato's generalized elimination rules in natural deduction [12] has been introduced and its simply-typed version proven to be strongly normalizing. Confluence and standardization hold even for the untyped calculus [6].

Since the aim of [5] has been to demonstrate the virtues of an approach without logical predicates/computability predicates for proving strong normalization of extensions of simply-typed lambda-calculus, it is natural to ask whether it can also prove strong normalization for any term which is strongly normalizing. Clearly, the answer to this question needs a type discipline which is complete in the sense that every strongly normalizing term is typable. It is well-known (see e. g. [7]) that in the case of λ-calculus this is true for an intersection type system without ω.

The aim of this article is to show a similar result for ΛJ. For this, the appropriate notion of intersection type assignment system has to be given, its completeness to be proved, and strong normalization to be established. Two proofs of strong normalization are given. The first one uses saturated sets for a Tait-style proof and shows that permutative conversions (which are naturally present in ΛJ) may also smoothly be integrated into such a proof. The second proof answers the above question positively for ΛJ: The proof method of [5] carries over to the intersection type system for ΛJ with only a small amount of extra effort in the reasoning.

In the next section, the extended λ-calculus ΛJ is defined and its strongly normalizing terms characterized by a set SN which is given by a syntax-directed inductive definition. In section 3 the right notion of intersection type assignment is given and its main properties established: closure under substitution and inverse substitution, inversion and subject reduction. Section 4 shows that every term in SN is typable while section 5 disproves this for a more naive notion of intersection typing. Strong normalization is shown in section 6 first in Tait-style (subsections 6.1 to 6.3) and then in subsection 6.4 in the style of [5].

Acknowledgements to Andreas Abel, Thorsten Altenkirch and Felix Joachimski for careful reading of parts and/or helpful suggestions.

2 A λ-Calculus with Generalized Applications

It is assumed that the reader knows λ-calculus (see e. g. [1, 7]).

The terms of ΛJ are generated from infinitely many variables, typically denoted by x,y,z, by λ-abstraction and generalized application: Every variable is a term; if r is a term then λxr is a term, and if r,s,t are terms then $r(s,z.t)$ is a term. The variable z gets bound in t (in the generalized application) as well as x in r (in lambda abstraction). By setting $rs := r(s,z.z)$, λ-calculus embeds into ΛJ. Terms which differ only in names of bound variables are identified, i.e., α-equal terms *are* equal. The capture-free *simultaneous substitution* of terms $\vec{s}$ for variables $\vec{x}$ in r may hence be defined as expected and will be written $r[\vec{x} := \vec{s}]$. Let $\mathrm{FV}(r)$ be the set of variables which occur free in r.

The letter S shall always denote syntactic objects of the form $(s,z.t)$. Lists $\vec{S}$ are used to denote multiple generalized applications: $r\vec{S} := rS_1 \dots S_n$ for $\vec{S} = S_1, \dots, S_n$.

Every term has one of the following shapes:

$$x \qquad xS \qquad xSS'\vec{S} \qquad \lambda xr \qquad (\lambda xr)S \qquad (\lambda xr)SS'\vec{S}.$$

The third and sixth forms are considered to be reducible by a permutative reduction since there are at least two subsequent generalized applications, and the fifth and sixth are reducible by β-reduction since an abstraction is followed by a generalized application. The reduction rules are [5, 6]:

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

$$r(s,z.t)S \to_\pi r(s,z.tS)$$

The *one-step reduction relation* $\to$ is generated from these conversions by means of the *compatible closure*, defined by

- If $r \to_\beta r'$ or $r \to_\pi r'$ then $r \to r'$.
- If $r \to r'$ then $r(s,z.t) \to r'(s,z.t)$, $s(r,z.t) \to s(r',z.t)$, $s(t,z.r) \to s(t,z.r')$ and $\lambda xr \to \lambda xr'$.

ΛJ can be embedded into λ-calculus by setting $r(s,z.t) := t[z := rs]$. If $r \to r'$ then the first translated term reduces in finitely many β-reduction steps of λ-calculus to the translation of r'. Note, however, that this does not help in proving strong normalization of ΛJ since, e. g., permutative steps are never simulated (by at least one reduction step in λ-calculus).

It is clear that the set NF of terms r for which there is no r' such that $r \to r'$ (called *normal forms*) is inductively characterized by

$$\frac{}{x \in \mathrm{NF}} \qquad \frac{s,t \in \mathrm{NF}}{x(s,z.t) \in \mathrm{NF}} \qquad \frac{r \in \mathrm{NF}}{\lambda xr \in \mathrm{NF}}$$

If this were simply-typed as

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

the rules would exactly correspond to the variable rule, the left $\to$-introduction rule and the right $\to$-introduction rule of cut-free Gentzen (sequent) calculus LJ, respectively. Since the set of all terms is derived from NF by substituting arbitrary terms for variables (apparently needed for the "main premise" x in the second rule), one may see ΛJ as the closure of the computational (i. e., type-free) content of cut-free LJ under substitution—hence the name ΛJ.

A term r is *strongly normalizing*, if there is no infinite reduction sequence originating in r, i. e. if it impossible that $r \to r_1 \to r_2 \to \dots$

The set SN is inductively defined by the following rules [5]:

$$\frac{}{x \in \mathsf{SN}} \qquad \frac{s,t \in \mathsf{SN}}{x(s,z.t) \in \mathsf{SN}}\,(\mathrm{V})$$

$$\frac{r \in \mathsf{SN}}{\lambda xr \in \mathsf{SN}}\,(\lambda) \qquad \frac{x(s,z.tS)\vec{S} \in \mathsf{SN}}{x(s,z.t)S\vec{S} \in \mathsf{SN}}\,(\pi)$$

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

It is shown in [5] that SN exactly captures the strongly normalizing terms of ΛJ.[1] This will be heavily used throughout this article, and it is motivated by a similar characterization for usual λ-calculus in [4, 11].

The above-mentioned embedding of ΛJ into λ-calculus justifies another rule: $r(s, z.t) \to_\kappa t$ if $z \notin \mathrm{FV}(t)$. Let us extend $\to$ appropriately. It is shown in [5] that SN also characterizes the strongly normalizing terms w. r. t. this extended reduction relation. In the following we assume that κ-reduction is included in our definition of ΛJ. However, this does not matter since the only reduction analysis happens in the proof of subject reduction. And clearly, subject reduction still holds if the reduction relation is restricted. But in order to understand from where Lemma 8 is coming, it is useful to think of κ being present in ΛJ.

3 Intersection Type Assignment

The intersection type assignment is that of system $\mathcal{D}$ (see e. g. [7, III2]) but with a different application rule. This rule is more liberal than the immediate generalization of the usual rule

$$\frac{\Gamma \vdash r : \rho \to \sigma \qquad \Gamma \vdash s : \rho}{\Gamma \vdash rs : \sigma}$$

for application in λ-calculus. In section 5 it will be shown that the liberality is essential for the result.

Let the set of types be generated from a fixed set of base types (which are typically denoted by ι) by using the binary connectives $\to$ and $\cap$, i. e. the types (typically denoted by ρ, σ, τ) are given inductively by: ι, $\rho \to \sigma$, $\rho \cap \sigma$.

From the following rules it becomes clear that only finite lists of variable names with types (each pair separated by a colon) may occur on the left-hand side of $\vdash$. We moreover restrict those *contexts* to lists with pairwise disjoint variable names. Hence, by writing $\Gamma, x : \rho$, it is implicit that x does not occur in Γ, written $x \notin \Gamma$. We also syntactically identify contexts which are permutations of each other. This amounts to using the following exchange rule without including it into the typing system: If Δ is a permutation of the pairs in Γ and $\Gamma \vdash r : \rho$ then $\Delta \vdash r : \rho$ (Exchange 1).[2]

[1] The fact that the strongly normalizing terms are contained in SN may be proved by main induction on the reduction tree height and side induction on the term structure, the other direction by proving that the strongly normalizing terms have all the closure properties which define SN.

[2] This rule would be derivable if the variable rule were extended to: $(x : \rho) \in \Gamma \Rightarrow \Gamma \vdash x : \rho$.

The relation $\Gamma \vdash r : \rho$ (term r has type ρ in context Γ) is inductively defined by the following rules:

$$\frac{}{\Gamma, x : \rho \vdash x : \rho}\,(\mathrm{V}) \qquad \frac{\Gamma, x : \rho \vdash r : \sigma}{\Gamma \vdash \lambda x r : \rho \to \sigma}\,(\lambda)$$

$$\frac{\Gamma \vdash r : \rho_1 \qquad \Gamma \vdash r : \rho_2}{\Gamma \vdash r : \rho_1 \cap \rho_2}\,(\cap\text{-I}) \qquad \frac{\Gamma \vdash r : \rho_1 \cap \rho_2 \qquad i \in \{1,2\}}{\Gamma \vdash r : \rho_i}\,(\cap\text{-E})$$

$$\frac{\Gamma \vdash r : \rho \qquad \Gamma \vdash s : \rho_1 \qquad \Gamma, z : \rho_2 \vdash t : \tau \qquad z \in \mathrm{FV}(t) \Rightarrow \rho = \rho_1 \to \rho_2}{\Gamma \vdash r(s, z.t) : \tau}\,(\mathrm{app})$$

Rule (app) may be seen as a shorthand for the following two rules:

$$\frac{\Gamma \vdash r : \rho_1 \to \rho_2 \qquad \Gamma \vdash s : \rho_1 \qquad \Gamma, z : \rho_2 \vdash t : \tau}{\Gamma \vdash r(s, z.t) : \tau}\,(\mathrm{app}_1)$$

$$\frac{\Gamma \vdash r : \rho \qquad \Gamma \vdash s : \sigma \qquad \Gamma \vdash t : \tau \qquad z \notin \mathrm{FV}(t)}{\Gamma \vdash r(s, z.t) : \tau}\,(\mathrm{app}_2)$$

Rule (app_1) is the obvious generalization of the application rule for λ-calculus. And since weakening and strengthening (see below) are admissible (in both formulations), rule (app_2) is equivalent to (app) in case $z \notin \mathrm{FV}(t)$.

Lemma 1 (Derived rules)

- *If* $\Gamma \vdash r : \rho$ *then* $\Gamma, x : \sigma \vdash r : \rho$ *(Weakening 1).*
- *If* $\Gamma, x : \sigma \vdash r : \rho$ *then* $\Gamma, x : \sigma \cap \tau \vdash r : \rho$ *(Weakening 2).*
- *If* $\Gamma, x : \sigma \vdash r : \rho$ *and* $x \notin \mathrm{FV}(r)$ *then* $\Gamma \vdash r : \rho$ *(Strengthening).*
- *If* $\Gamma, x : \rho \cap \sigma \vdash r : \tau$ *then* $\Gamma, x : \sigma \cap \rho \vdash r : \tau$ *(Exchange 2).*
- *If* $\Gamma, x : (\rho_1 \cap \rho_2) \cap \rho_3 \vdash r : \tau$ *then* $\Gamma, x : \rho_1 \cap (\rho_2 \cap \rho_3) \vdash r : \tau$ *(Exchange 3).*

Proof By rule induction on $\vdash$. □

Lemma 2 (Substitution)

(a) *If* $\Gamma, x : \rho \vdash r : \tau$ *and* $\Gamma \vdash s : \rho$ *then* $\Gamma \vdash r[x := s] : \tau$.

(b) *If* $\Gamma \vdash r[x := s] : \tau$, $x \notin \mathrm{FV}(s) \cup \Gamma$ *and* $(x \notin \mathrm{FV}(r) \Rightarrow \Gamma \vdash s : \rho_0)$ *then there is a type* ρ *such that* $\Gamma, x : \rho \vdash r : \tau$ *and* $\Gamma \vdash s : \rho$.

Proof (a) has a completely straightforward proof by induction on the derivation of $\Gamma, x : \rho \vdash r : \tau$ and does not reveal anything specific to intersection types. (b) is typical of intersection types (compare with Proposition 3 in [7, p. 51] which is unfortunately too weak since it always requires $\Gamma \vdash s : \rho_0$ for some type ρ_0) and is proved by induction on the derivation of $\Gamma \vdash r[x := s] : \tau$. The case $r = x$ is trivial. Let now $r \neq x$. Since the arguments are standard for the other cases, we only consider rule (app). Then $\Gamma \vdash r[x := s](s'[x := s], z.t[x := s]) : \tau$ due to $\Gamma \vdash r[x := s] : \sigma$, $\Gamma \vdash s'[x := s] : \sigma_1$ and $\Gamma, z : \sigma_2 \vdash t[x := s] : \tau$ (we may assume that $z \notin \{x\} \cup \mathrm{FV}(s)$) and $z \in \mathrm{FV}(t[x := s]) \Rightarrow \sigma = \sigma_1 \to \sigma_2$. If $x \in \mathrm{FV}(r(s', z.t))$ then $x \in \mathrm{FV}(r) \cup \mathrm{FV}(s') \cup \mathrm{FV}(t)$. In each case the induction hypothesis provides a type ρ_0 such that $\Gamma \vdash s : \rho_0$ (in the third case use Strengthening due to $z \notin \mathrm{FV}(s)$). Hence, from the assumption in (b) we always have some ρ_0 with $\Gamma \vdash s : \rho_0$. By Weakening 1, also $\Gamma, z : \sigma_2 \vdash s : \rho_0$. By induction hypothesis, there are types ρ_1, ρ_2, ρ_3 such that $\Gamma, x : \rho_1 \vdash r : \sigma$, $\Gamma \vdash s : \rho_1$, $\Gamma, x : \rho_2 \vdash s' : \sigma_1$, $\Gamma \vdash s : \rho_2$, $\Gamma, z : \sigma_2, x : \rho_3 \vdash t : \tau$ and $\Gamma, z : \sigma_2 \vdash s : \rho_3$. By Strengthening, $\Gamma \vdash s : \rho_3$. Since $z \in \mathrm{FV}(t)$ implies $z \in \mathrm{FV}(t[x := s])$ which in turn yields $\sigma = \sigma_1 \to \sigma_2$, Weakening 2 and Exchange 2 allow us to conclude $\Gamma, x : (\rho_1 \cap \rho_2) \cap \rho_3 \vdash r(s', z.t) : \tau$. Finally, $\Gamma \vdash s : (\rho_1 \cap \rho_2) \cap \rho_3$. □

The next lemma will always be needed if a given type assignment is analyzed. For its statement we need the concepts of prime types and prime factors [7, p. 50]: A *prime type* is a type which is not of the form $\rho \cap \sigma$. Let $\mathcal{P}$ be the set of prime types. Every type can be written in the form $\rho_1 \cap \ldots \cap \rho_n$ with $n \geq 1$ and $\rho_i \in \mathcal{P}$ for $i \in \{1, \ldots, n\}$. Note that we do not care about parentheses since they do not change typability (neither in contexts due to Exchange 3 nor in the derived type due to ($\cap$-I) and ($\cap$-E)). The ρ_i are called *prime factors* of ρ. We now adapt Lemma 1 from [7, p. 50] to ΛJ. In the sequel the notation $\cap\vec{\rho}$ will be used for $\rho_1 \cap \ldots \cap \rho_n$ with the implicit assumption that the ρ_i are prime types. $\rho \in \vec{\rho}$ clearly means that $\rho = \rho_i$ for some i.

Lemma 3 (Inversion) *Let* $\Gamma \vdash r : \rho$ *with* $\rho \in \mathcal{P}$.

1. *If* $r = x$ *then* $x : \cap\vec{\rho} \in \Gamma$ *with* $\rho \in \vec{\rho}$.
2. *If* $r = \lambda xt$ *then* $\rho = \rho_1 \to \rho_2$ *and* $\Gamma, x : \rho_1 \vdash t : \rho_2$.[3]
3. *If* $r = r'(s, z.t)$ *then* $\Gamma \vdash r' : \sigma$, $\Gamma \vdash s : \sigma_1$, $\Gamma, z : \sigma_2 \vdash t : \rho$ *and* $(z \in \mathrm{FV}(t) \Rightarrow \sigma = \sigma_1 \to \sigma_2)$.

Proof Consider in the derivation of $\Gamma \vdash r : \rho$ the first occurrence of some $\Gamma \vdash r : \cap\vec{\rho}$ with $\rho \in \vec{\rho}$. The rule by which this is achieved cannot be ($\cap$-I) or ($\cap$-E) since they would require an earlier occurrence of the described form. Therefore, in the first and second case, the rule is (V) and (λ), respectively. In the third case, it is (app) and therefore $\Gamma, z : \sigma_2 \vdash t : \cap\vec{\rho}$, hence $\Gamma, z : \sigma_2 \vdash t : \rho$. □

[3] We may assume that $x \notin \Gamma$. The same holds of z in the next case.

Theorem 1 (Subject reduction) *If* $\Gamma \vdash r : \rho$ *and* $r \to r'$ *then* $\Gamma \vdash r' : \rho$.

Proof By induction on $\Gamma \vdash r : \rho$. Only the case (app) is non-trivial: We have $\Gamma \vdash r(s,z.t) : \tau$ due to $\Gamma \vdash r : \rho$, $\Gamma \vdash s : \rho_1$, $\Gamma, z : \rho_2 \vdash t : \tau$, $(z \in \mathrm{FV}(t) \Rightarrow \rho = \rho_1 \to \rho_2)$, and $r(s,z.t) \to v$. We have to show that $\Gamma \vdash v : \tau$.

- If $v = r'(s',z.t')$ by one reduction step altogether from r to r', s to s' and t to t' then we are done by induction hypothesis and $\mathrm{FV}(t') \subseteq \mathrm{FV}(t)$.

- In the case of an outer κ-reduction $z \notin \mathrm{FV}(t)$ and $v = t$. By Strengthening, $\Gamma \vdash t : \tau$.

- In the case of an outer β-reduction $r = \lambda x r_0$ and $v = t[z := r_0[x := s]]$. If $z \notin \mathrm{FV}(t)$ this is an outer κ-reduction. Otherwise, $\rho = \rho_1 \to \rho_2$. By the preceding lemma, $\Gamma, x : \rho_1 \vdash r_0 : \rho_2$. By Lemma 2(a), $\Gamma \vdash r_0[x := s] : \rho_2$. Again by Lemma 2(a), $\Gamma \vdash v : \tau$.

- Finally, in the case of an outer permutative reduction, $r = r_0(s_0, z_0.t_0)$ and $v = r_0(s_0, z_0.t_0(s,z.t))$. $\rho = \cap\vec{\sigma}$. Therefore, $\Gamma \vdash r_0(s_0,z_0.t_0) : \sigma_1$ (the first element of $\vec{\sigma}$) and by the preceding lemma, $\Gamma \vdash r_0 : \rho'$, $\Gamma \vdash s_0 : \rho'_1$, $\Gamma, z_0 : \rho'_2 \vdash t_0 : \sigma_1$ and $(z_0 \in \mathrm{FV}(t_0) \Rightarrow \rho' = \rho'_1 \to \rho'_2)$. By Weakening 1, $\Gamma, z_0 : \rho'_2 \vdash s : \rho_1$ and $\Gamma, z_0 : \rho'_2, z : \rho_2 \vdash t : \tau$. If $z \in \mathrm{FV}(t)$ then $\rho = \rho_1 \to \rho_2$, hence also $\sigma_1 = \rho$ (only one prime factor). By (app), we conclude $\Gamma, z_0 : \rho'_2 \vdash t_0(s,z.t) : \tau$. Since, by the implicit proviso of permutative reductions, $z_0 \in \mathrm{FV}(t_0(s,z.t))$ implies $z_0 \in \mathrm{FV}(t_0)$ which in turn implies $\rho' = \rho'_1 \to \rho'_2$, a final application of rule (app) yields $\Gamma \vdash v : \tau$. □

4 Completeness of Intersection Type Assignment

We show that every strongly normalizing term of ΛJ is typable, more precisely, it is shown that the intersection typing system captures the set SN completely. This is done by showing that the set of typable terms has all the closure properties defining SN. First we consider the most delicate subcases.

Lemma 4 *If* $\Gamma \vdash t[z := r[x := s]] : \tau$, $\Gamma, x : \sigma' \vdash r : \rho$ *and* $\Gamma \vdash s : \sigma$ *then* $\Gamma \vdash (\lambda x r)(s,z.t) : \tau$.

Proof We may assume that $x \notin \mathrm{FV}(s)$ and $z \notin \mathrm{FV}(r[x := s]) \cup \Gamma$. If $z \notin \mathrm{FV}(t)$ then $\Gamma \vdash \lambda x r : \sigma' \to \rho$ and, by Weakening 1, $\Gamma, z : \rho' \vdash t : \tau$ for some arbitrary type ρ'. Therefore, $\Gamma \vdash (\lambda x r)(s,z.t) : \tau$. Otherwise, by Lemma 2(b), there is a type ρ' such that $\Gamma, z : \rho' \vdash t : \tau$ and $\Gamma \vdash r[x := s] : \rho'$. Again by Lemma 2(b), there is some σ'' with $\Gamma, x : \sigma'' \vdash r : \rho'$ and $\Gamma \vdash s : \sigma''$. Therefore $\Gamma \vdash \lambda x r : \sigma'' \to \rho'$ and finally $\Gamma \vdash (\lambda x r)(s,z.t) : \tau$. □

Lemma 5 *If* $\Gamma \vdash r(s,z.tS) : \tau$ *then* $\Gamma \vdash r(s,z.t)S : \tau$.

Proof It clearly suffices to show the lemma for $\tau \in \mathcal{P}$. Hence, let $\tau \in \mathcal{P}$. By Lemma 3, $\Gamma \vdash r : \rho$, $\Gamma \vdash s : \rho_1$, $\Gamma, z : \rho_2 \vdash tS : \tau$ and $(z \in \mathrm{FV}(tS) \Rightarrow \rho = \rho_1 \to \rho_2)$. Let $S = (s', z'.t')$. Again by Lemma 3, we have $\Gamma, z : \rho_2 \vdash t : \rho'$, $\Gamma, z : \rho_2 \vdash s' : \rho'_1$, $\Gamma, z : \rho_2, z' : \rho'_2 \vdash t' : \tau$ and $(z' \in \mathrm{FV}(t') \Rightarrow \rho' = \rho'_1 \to \rho'_2)$. Since $z \in \mathrm{FV}(t)$ implies $z \in \mathrm{FV}(tS)$ which in turn implies $\rho = \rho_1 \to \rho_2$, rule (app) yields $\Gamma \vdash r(s, z.t) : \rho'$. By Strengthening (due to the implicit proviso that $z \notin \mathrm{FV}(s') \cup \mathrm{FV}(t')$), $\Gamma \vdash s' : \rho'_1$ and $\Gamma, z' : \rho'_2 \vdash t' : \tau$. Rule (app) is applicable and gives the result. □

Let us write $\Gamma + \Delta$ for the context where the requirements on the variables in Γ and Δ are added. More formally, if $\Gamma = \vec{x} : \vec{\rho}, \vec{y} : \vec{\sigma}$ and $\Delta = \vec{x} : \vec{\rho}\,', \vec{z} : \vec{\tau}$ with $\vec{x}$, $\vec{y}$ and $\vec{z}$ pairwise disjoint and $\vec{x}$ of length n, then $\Gamma + \Delta := x_1 : \rho_1 \cap \rho'_1, \ldots, x_n : \rho_n \cap \rho'_n, \vec{y} : \vec{\sigma}, \vec{z} : \vec{\tau}$. From Weakening 1 and Weakening 2, it follows that $\Gamma \vdash r : \rho$ implies $\Gamma + \Delta \vdash r : \rho$.

Theorem 2 (Completeness) *If $r \in \mathsf{SN}$ then there are Γ and ρ such that $\Gamma \vdash r : \rho$.*

Proof Induction on SN. (V) and (λ) are treated by the induction hypothesis and some weakening.

(π). Let $x(s, z.t)S\vec{S} \in \mathsf{SN}$ be derived from $x(s, z.tS)\vec{S} \in \mathsf{SN}$. By induction hypothesis, $\Gamma \vdash x(s, z.tS)\vec{S} : \tau$. Writing $x(s, z.tS)\vec{S} = (y\vec{S})[y := x(s, z.tS)]$ with fresh variable y, we conclude by Lemma 2(b) that $\Gamma, y : \rho \vdash y\vec{S} : \tau$ and $\Gamma \vdash x(s, z.tS) : \rho$ for some type ρ. By the previous lemma, $\Gamma \vdash x(s, z.t)S : \rho$. Therefore, by Lemma 2(a), $\Gamma \vdash (y\vec{S})[y := x(s, z.t)S] : \tau$.

(β). By a similar argument, some weakening and Lemma 4. □

5 Incompleteness without (app$_2$)

We show that it is essential to allow in (app) that for $z \notin \mathrm{FV}(t)$, the types of r, s and z need not match.

Lemma 6 *Let $\vec{\rho} \subset \mathcal{P}$. The following is impossible:*

$$\Gamma \vdash \lambda x.x(x, z.z) : \cap\vec{\rho} \to \sigma \text{ and } \Gamma \vdash \lambda x.x(x, z.z) : \cap\vec{\rho}.$$

Proof Induction on the number of different types in $\vec{\rho}$. Assume both typings. By Lemma 3, $\Gamma, x : \cap\vec{\rho} \vdash x(x, z.z) : \sigma$. $\sigma = \cap\vec{\sigma}$. Therefore, $\Gamma, x : \cap\vec{\rho} \vdash x(x, z.z) : \sigma_1$ (the first element of $\vec{\sigma}$). Again by Lemma 3, $\Gamma, x : \cap\vec{\rho} \vdash x : \tau \to \sigma'$ and $\Gamma, x : \cap\vec{\rho} \vdash x : \tau$. τ decomposes into prime factors as $\tau = \tau_1 \cap \ldots \cap \tau_n$, hence for every $k \in \{1, \ldots, n\}$: $\Gamma, x : \cap\vec{\rho} \vdash x : \tau_k$. Again by Inversion, $\tau_k \in \vec{\rho}$ for every $k \in \{1, \ldots, n\}$. Therefore, $\tau = \cap\vec{\rho}\,'$ for some finite list $\vec{\rho}\,'$ composed only of elements of $\vec{\rho}$. Once more by Inversion, applied to $\Gamma, x : \cap\vec{\rho} \vdash x : \cap\vec{\rho}\,' \to \sigma'$, there is a ρ_j in $\vec{\rho}$ such that $\rho_j = \cap\vec{\rho}\,' \to \sigma'$. Hence $\rho_j \notin \vec{\rho}\,'$, and consequently $\vec{\rho}\,'$ has fewer different types than $\vec{\rho}$. From $\Gamma \vdash \lambda x.x(x, z.z) : \cap\vec{\rho}$ we now immediately get $\Gamma \vdash \lambda x.x(x, z.z) : \cap\vec{\rho}\,' \to \sigma'$ and $\Gamma \vdash \lambda x.x(x, z.z) : \cap\vec{\rho}\,'$. Contradiction by induction hypothesis. □

Now **assume** until the end of this section **that (app) is replaced** by (app_2). It is clear that Lemma 1 remains valid. Lemma 3 clearly has to be changed in case 3 to: *If* $r = r'(s, z.t)$ *then* $\Gamma \vdash r' : \sigma_1 \to \sigma_2$, $\Gamma \vdash s : \sigma_1$ *and* $\Gamma, z : \sigma_2 \vdash t : \rho$. Call the new lemma also Inversion. It is proved as before. Since the typing rules are now more restricted, clearly also the previous lemma remains valid.

Lemma 7 $t := (\lambda x.x(x, z.z))(\lambda x.x(x, z.z), y.z) \in \mathsf{SN}$ *but not typable in the restricted intersection typing system.*

Proof It is even obvious that t is strongly normalizing (its only reduct is z). Assume $\Gamma \vdash t : \tau$. We may assume that $\tau \in \mathcal{P}$. By our variant of Inversion, $\Gamma \vdash \lambda x.x(x, z.z) : \rho \to \sigma$ and $\Gamma \vdash \lambda x.x(x, z.z) : \rho$. Since $\rho = \cap\vec{\rho}$, the lemma applies. $\square$

A much easier proof can be given which uses that every typable term is in SN (to be proved in the next section): Again assume $\Gamma \vdash t : \tau$ with $\tau \in \mathcal{P}$. The variant of Inversion again yields $\Gamma \vdash \lambda x.x(x, z.z) : \rho \to \sigma$, $\Gamma \vdash \lambda x.x(x, z.z) : \rho$ and $\Gamma, y : \sigma \vdash z : \tau$. Trivially, also $\Gamma, y : \sigma \vdash y : \sigma$. Hence $\Gamma \vdash (\lambda x.x(x, z.z))(\lambda x.x(x, z.z), y.y) : \sigma$. By soundness (which will even be proved for the full typing system),

$$(\lambda x.x(x, z.z))(\lambda x.x(x, z.z), y.y) \in \mathsf{SN}$$

which is clearly false (the term is obviously not strongly normalizing since it is the $\Lambda\mathrm{J}$ term corresponding to $(\lambda x.xx)(\lambda x.xx)$ in λ-calculus).

6 Soundness of Intersection Type Assignment

It is proved that every typable term is strongly normalizing. More precisely, it is shown that all typable terms are contained in SN. The first proof is by the computability predicates method employing saturated sets in the spirit of [9]. We will closely follow[4] the exposition in [8, pp. 114–127] which only dealt with β-reduction and no intersection types but successfully carried over even to systems of monotone inductive types (being the theme of [8]). It is well-known that the computability predicates method is perfectly capable of dealing with intersection types [7, 10, 3]. The main aim is to show that even the addition of permutative conversions does not produce any noteworthy overhead. Given that one has established the inductive characterization SN of the strongly normalizing terms, there are merely two places in the proof which need an argument concerning (π) not being uniformly treated together with (β).

The second proof is the extension of that for simply-typed $\Lambda\mathrm{J}$ from [5] to the intersection type system. That proof does not use any kind of computability predicates but only proves closure of SN under generalized application and under

[4] The completely different notation is due to Felix Joachimski.

substitution by an outer induction on types (only typed terms are considered) and an inner induction on the inductive definition of SN.[5]

6.1 Definition of Saturatedness and Saturated Closure

A set $\mathcal{M}$ of terms is called *saturated* (for λ-calculus see e. g. [1, p. 177]), if the following conditions are met:

(SN) $\mathcal{M} \subseteq \mathsf{SN}$.

(V) $x \in \mathcal{M}$, and if $s \in \mathsf{SN}$ and $t \in \mathcal{M}$ then $x(s,z.t) \in \mathcal{M}$.

(π) If $x(s,z.tS)\vec{S} \in \mathcal{M}$ then $x(s,z.t)S\vec{S} \in \mathcal{M}$.

(β) If $t[z := r[x := s]]\vec{S} \in \mathcal{M}$ and $r,s \in \mathsf{SN}$ then $(\lambda xr)(s,z.t)\vec{S} \in \mathcal{M}$.

The transformation of the definition of SN into that of saturatedness is well motivated by typing considerations: It would be possible to relativize the definition to τ-saturatedness for a type τ by restricting to terms of type τ only. For this to work it is essential to have no requirement in the rules that a term of a different type shall be in $\mathcal{M}$ in order to conclude that some term belongs to $\mathcal{M}$. Therefore we omit the rule which changes the type (hence leave out (λ)), and in the other rules replace every occurrence of SN by $\mathcal{M}$ in the conclusion and for the terms in the premisses receiving the same type. In general, the other terms do not get the same type and therefore can only be required to belong to SN. In order to deal with permutative conversions, it is essential to have $t \in \mathcal{M}$ in the premise of (V) as will become clear in the proof of part (2) of Lemma 9. This also fits with the relativization to a typed version since t and $x(s,z.t)$ always receive the same type.

Lemma 8 (Derived κ-rule) *Let $\mathcal{M}$ be saturated. Then we have the following κ-expansion rule: If $r,s \in \mathsf{SN}$, $t \in \mathcal{M}$ and $z \notin \mathrm{FV}(t)$ then $r(s,z.t) \in \mathcal{M}$.*

Proof Induction on $r \in \mathsf{SN}$. Only (λ) uses that $z \notin \mathrm{FV}(t)$. □

Let SAT be the set of saturated sets. Trivially, $\mathsf{SN} \in \mathsf{SAT}$.

It is always possible to produce a saturated set from any set M of terms by the *saturated closure* $\mathsf{cl}(M)$, defined by induction as follows:

($\subseteq$) If $r \in M \cap \mathsf{SN}$ then $r \in \mathsf{cl}(M)$.

(V) $x \in \mathsf{cl}(M)$, and if $s \in \mathsf{SN}$ and $t \in \mathsf{cl}(M)$ then $x(s,z.t) \in \mathsf{cl}(M)$.

(π) If $x(s,z.tS)\vec{S} \in \mathsf{cl}(M)$ then $x(s,z.t)S\vec{S} \in \mathsf{cl}(M)$.

[5] There is a fly in the ointment, though: One has to show that SN is not only closed under (π) which it is by definition, but even under: $r(s,z.tS)\vec{S} \in \mathsf{SN} \Rightarrow r(s,z.t)S\vec{S} \in \mathsf{SN}$ for arbitary terms r. This is some implicit form of standardization for permutative conversions the need for which is avoided in the first proof.

(β) If $t[z := r[x := s]]\vec{S} \in \mathsf{cl}(M)$ and $r, s \in \mathsf{SN}$ then $(\lambda xr)(s, z.t)\vec{S} \in \mathsf{cl}(M)$.

Since $\mathsf{cl}(M) \subseteq \mathsf{SN}$ (proved by induction on the definition), it is the least saturated set containing $M \cap \mathsf{SN}$.[6]

6.2 Calculating with Saturated Sets

Given a saturated set $\mathcal{M}$ and a saturated set $\mathcal{N}$, we want to construct a saturated set $\mathcal{M} \to \mathcal{N}$ which will later serve to define strong computability for function types.

Let $\mathcal{M}$ and $\mathcal{N}$ be (arbitrary) sets of terms. Define

$$\mathsf{S}_x(\mathcal{M}, \mathcal{N}) := \{r \in \mathcal{N} \cap \mathsf{SN} \,|\, \forall s \in \mathcal{M}\, r[x := s] \in \mathcal{N}\},$$

$$\mathsf{I}(\mathcal{M}, \mathcal{N}) := \{\lambda xr \,|\, x \text{ variable and } r \in \mathsf{S}_x(\mathcal{M}, \mathcal{N})\} \quad \text{and}$$

$$\mathsf{E}(\mathcal{M}, \mathcal{N}) := \{r \,|\, \forall s \in \mathcal{M} \cap \mathsf{SN}\, \forall \mathcal{P} \in \mathsf{SAT}\, \forall z \forall t \in \mathsf{S}_z(\mathcal{N}, \mathcal{P})\, r(s, z.t) \in \mathcal{P}\}.$$

We get the introduction-based definition $\mathcal{M} \to_I \mathcal{N}$ and the elimination-based definition $\mathcal{M} \to_E \mathcal{N}$ of saturated sets:

$$\mathcal{M} \to_I \mathcal{N} := \mathsf{cl}(\mathsf{I}(\mathcal{M}, \mathcal{N})) \text{ and } \mathcal{M} \to_E \mathcal{N} := \mathsf{cl}(\mathsf{E}(\mathcal{M}, \mathcal{N})).$$

Lemma 9 $\mathsf{I}(\mathcal{M}, \mathcal{N}) \subseteq \mathsf{SN}$, $\mathsf{E}(\mathcal{M}, \mathcal{N}) \cap \mathsf{SN} \in \mathsf{SAT}$, *and* $\mathsf{I}(\mathcal{M}, \mathcal{N}) \subseteq \mathsf{E}(\mathcal{M}, \mathcal{N})$.

Proof

(1) By rule (λ) of SN.

(2) Check the conditions of saturatedness for $\mathsf{E}(\mathcal{M}, \mathcal{N}) \cap \mathsf{SN}$:

(SN). Trivial.

(V). $x \in \mathsf{E}(\mathcal{M}, \mathcal{N})$: Let $s, \mathcal{P}, z, t$ as specified in the definition of $\mathsf{E}(\mathcal{M}, \mathcal{N})$. In order to show $x(s, z.t) \in \mathcal{P}$, we need $s \in \mathsf{SN}$ and $t \in \mathcal{P}$ which are both given.

Let $s' \in \mathsf{SN}$, $t' \in \mathsf{E}(\mathcal{M}, \mathcal{N}) \cap \mathsf{SN}$. Show that $x(s', z'.t') \in \mathsf{E}(\mathcal{M}, \mathcal{N}) \cap \mathsf{SN}$. $x(s', z'.t') \in \mathsf{SN}$ is obvious. Let again $s, \mathcal{P}, z, t$ be as specified in the definition of $\mathsf{E}(\mathcal{M}, \mathcal{N})$. We have to show that $x(s', z'.t')(s, z.t) \in \mathcal{P}$. By rule (π)[7], it suffices to show $x(s', z'.t'(s, z.t)) \in \mathcal{P}$. By rule (V), we only need $s' \in \mathsf{SN}$ and $t'(s, z.t) \in \mathcal{P}$ which clearly follows from $t' \in \mathsf{E}(\mathcal{M}, \mathcal{N})$.

[6] Of course, we could give a fully impredicative justification of the existence of such a set: SAT is a complete lattice since it is closed under arbitrary intersections. However, $\mathsf{cl}(M)$ only needs strictly-positive inductive definitions.

[7] At no other place except case (V) in the proof of Lemma 8, we need to refer in this first normalization proof to the concrete definition of (π) since below we only use that (π) is closed under generalized application as expressed by the trailing vector of generalized applications.

(π), (β). Simply add the generalized argument $(s,z.t)$ to the multiple generalized application indicated by $\vec{S}$ in both rules and use saturatedness of $\mathcal{P}$.

(3) Let $r \in \mathsf{S}_x(\mathcal{M},\mathcal{N})$, $s \in \mathcal{M} \cap \mathsf{SN}$, $\mathcal{P} \in \mathsf{SAT}$ and $t \in \mathsf{S}_z(\mathcal{N},\mathcal{P})$. We have to show $(\lambda xr)(s,z.t) \in \mathcal{P}$. By ($\beta$), it suffices to show $t[z := r[x := s]] \in \mathcal{P}$ and $r,s \in \mathsf{SN}$. $r[x := s] \in \mathcal{N}$ by definition of $\mathsf{S}_x(\mathcal{M},\mathcal{N})$, hence $t[z := r[x := s]] \in \mathcal{P}$ by definition of $\mathsf{S}_z(\mathcal{N},\mathcal{P})$. $\square$

From the lemma, we get $\mathsf{I}(\mathcal{M},\mathcal{N}) \subseteq \mathcal{M} \to_I \mathcal{N}$, $\mathcal{M} \to_E \mathcal{N} = \mathsf{E}(\mathcal{M},\mathcal{N}) \cap \mathsf{SN}$ and, due to monotonicity of cl, $\mathcal{M} \to_I \mathcal{N} \subseteq \mathcal{M} \to_E \mathcal{N}$.

Define $\mathcal{M} \to \mathcal{N} := \mathcal{M} \to_X \mathcal{N}$ with $X \in \{I,E\}$. We never use any property depending on this choice but only the following three properties which are valid for both choices and follow immediately from the preceding remarks:

(SAT) $\mathcal{M} \to \mathcal{N} \in \mathsf{SAT}$.

($\to$-I) If $r \in \mathsf{S}_x(\mathcal{M},\mathcal{N})$ then $\lambda xr \in \mathcal{M} \to \mathcal{N}$.

($\to$-E) $r \in \mathcal{M} \to \mathcal{N} \wedge s \in \mathcal{M} \cap \mathsf{SN} \wedge \mathcal{P} \in \mathsf{SAT} \wedge t \in \mathsf{S}_z(\mathcal{N},\mathcal{P}) \Rightarrow r(s,z.t) \in \mathcal{P}$.

Note that if $x \in \mathcal{M}$ and $\mathcal{N} \subseteq \mathsf{SN}$ then (by choosing $s := x \in \mathcal{M}$)

$$\mathsf{S}_x(\mathcal{M},\mathcal{N}) = \{r \mid \forall s \in \mathcal{M}\, r[x := s] \in \mathcal{N}\}.$$

As a consequence, if every variable is contained in $\mathcal{M}$ and $\mathcal{N}$ and $\mathcal{M},\mathcal{N} \subseteq \mathsf{SN}$, ($\to$-I) and ($\to$-E) may be rewritten to

($\to$-I) If $r[x := s] \in \mathcal{N}$ for all $s \in \mathcal{M}$ then $\lambda xr \in \mathcal{M} \to \mathcal{N}$.

($\to$-E) If $r \in \mathcal{M} \to \mathcal{N}$, $s \in \mathcal{M}$, $\mathcal{P} \in \mathsf{SAT}$ and $t[z := v] \in \mathcal{P}$ for every $v \in \mathcal{N}$ then $r(s,z.t) \in \mathcal{P}$.

Since from now on $\mathcal{M}$ and $\mathcal{N}$ will represent saturated sets, these latter rules apply.

6.3 Strong Computability

Define the saturated set SC^ρ of *strongly computable terms* w. r. t. type ρ by recursion on ρ:

(ι) $\mathsf{SC}^\iota := \mathsf{SN}$.

($\to$) $\mathsf{SC}^{\rho\to\sigma} := \mathsf{SC}^\rho \to \mathsf{SC}^\sigma$.

($\cap$) $\mathsf{SC}^{\rho\cap\sigma} := \mathsf{SC}^\rho \cap \mathsf{SC}^\sigma$ (with set-theoretic intersection).

We want to show that every typable term of ΛJ is strongly normalizing. By using the inductive characterization, we only need to show that they are in SN. Since saturated sets are contained in SN, it suffices to show that $r \in \mathsf{SC}^\rho$ whenever r gets type ρ. This is achieved by applying the following lemma to the identity substitution.

Lemma 10 *If* $\vec{x} : \vec{\rho} \vdash r : \rho$ *and* $\vec{s} \in \mathsf{SC}^{\vec{\rho}}$ *then* $r[\vec{x} := \vec{s}] \in \mathsf{SC}^\rho$.

Proof By induction on $\vec{x} : \vec{\rho} \vdash r : \rho$. Again, only (app) deserves interest: We have $\vec{x} : \vec{\rho} \vdash r(s,z.t) : \tau$ due to $\vec{x} : \vec{\rho} \vdash r : \rho$, $\vec{x} : \vec{\rho} \vdash s : \rho_1$, $\vec{x} : \vec{\rho}, z : \rho_2 \vdash t : \tau$ and $(z \in \mathrm{FV}(t) \Rightarrow \rho = \rho_1 \to \rho_2)$. We have to show that $r[\vec{x} := \vec{s}](s[\vec{x} := \vec{s}], z.t[\vec{x} := \vec{s}]) \in \mathsf{SC}^\tau$. If $z \in \mathrm{FV}(t)$ then by property ($\to$-E), we only have to show $r[\vec{x} := \vec{s}] \in \mathsf{SC}^{\rho_1 \to \rho_2}$, $s[\vec{x} := \vec{s}] \in \mathsf{SC}^{\rho_1}$ and for all $v \in \mathsf{SC}^{\rho_2}$: $t[\vec{x} := \vec{s}][z := v] = t[\vec{x}, z := \vec{s}, v] \in \mathsf{SC}^\tau$. This follows from the induction hypothesis. If $z \notin \mathrm{FV}(t)$ then by Lemma 8, it suffices to show $r[\vec{x} := \vec{s}] \in \mathsf{SN}$, $s[\vec{x} := \vec{s}] \in \mathsf{SN}$ and $t[\vec{x} := \vec{s}] = t[\vec{x}, z := \vec{s}, z] \in \mathsf{SC}^\tau$ which again follow from the induction hypothesis since $z \in \mathsf{SC}^{\rho_2}$ and $\mathsf{SC}^\rho, \mathsf{SC}^{\rho_1} \subseteq \mathsf{SN}$. □

By setting $s_i := x_i \in \mathsf{SC}^{\rho_i}$ for $\Gamma = \vec{x} : \vec{\rho}$, and by using $\mathsf{SN}^\rho \subseteq \mathsf{SN}$, we get the following

Theorem 3 (Soundness/Strong Normalization) *If* $\Gamma \vdash r : \rho$ *then* $r \in \mathsf{SN}$. □

6.4 Second Proof without Logical Predicates

The following proof does not use the hierarchy of definitions of strongly computable terms with ever increasing logical complexity. On the other hand, it does not seem to be as extensible as the first proof: Although [5] exhibits an extension of the second proof method to Gödel's system T (but without intersection types), it is not yet clear how to treat extensions by inductive types not having closed inhabitants in normal form of such an easy structure as the numerals in Gödel's T.

As already mentioned in footnote 5, we need another derived rule of SN.

Lemma 11 (Derived π-rule [5]) *If* $r(s,z.tS)\vec{S} \in \mathsf{SN}$ *then* $r(s,z.t)S\vec{S} \in \mathsf{SN}$.

Proof By course-of-generation induction on $r(s,z.tS)\vec{S} \in \mathsf{SN}$, see [5]. □

The following lemma extends the corresponding lemma in [5] from simply-typed ΛJ to the present ΛJ with intersection types.

Lemma 12 *For all types* ρ, *for all* $r \in \mathsf{SN}$,

(i) *if* $\Gamma \vdash r : \rho$, $\Gamma \vdash s : \rho_1$, $\Gamma, z : \rho_2 \vdash t : \tau$ *and* $s, t \in \mathsf{SN}$ *and* $(z \in \mathrm{FV}(t) \Rightarrow \rho = \rho_1 \to \rho_2)$ *then* $r(s,z.t) \in \mathsf{SN}$,

(ii) *if* $\Gamma, x : \rho \vdash r : \tau$, $\Gamma \vdash s : \rho$ *and* $s \in \mathsf{SN}$ *then* $r[x := s] \in \mathsf{SN}$.

Proof First remark that in case (i) we have $\Gamma \vdash r(s,z.t) : \tau$ and in case (ii) due to Lemma 2(a) $\Gamma \vdash r[x := s] : \tau$.
The proof is by simultaneous induction on ρ, side induction on $r \in \mathsf{SN}$, and case analysis according to $r \in \mathsf{SN}$. One always first has to prove (i) and later infers (ii), possibly with the help of (i). As in [5], the most interesting cases are (ii)(V) and (i)(λ). All the other cases do not use the main induction hypothesis and only use the side induction hypothesis for the same statement, i. e. (i) for (i) and (ii) for (ii).[8]

(ii)(V). $r = x$ and $r = y \neq x$ are trivial. Let $y(s',z'.t') \in \mathsf{SN}$ be derived from $s',t' \in \mathsf{SN}$. Show that $(y(s',z'.t'))[x := s] \in \mathsf{SN}$. By side induction hypothesis (ii), $s'[x := s], t'[x := s] \in \mathsf{SN}$. Hence, we are done if $y \neq x$. Otherwise show that $s(s'[x := s], z'.t'[x := s]) \in \mathsf{SN}$. $\tau = \cap\vec{\tau}$, hence from $\Gamma, x : \rho \vdash x(s',z'.t') : \tau$ we get $\Gamma, x : \rho \vdash x(s',z'.t') : \tau_1$ ($\vec{\tau}$'s first element). By Lemma 3, $\Gamma, x : \rho \vdash x : \rho'$, $\Gamma, x : \rho \vdash s' : \rho'_1$, $\Gamma, x : \rho, z' : \rho'_2 \vdash t' : \tau_1$ and $(z' \in \mathrm{FV}(t') \Rightarrow \rho' = \rho'_1 \to \rho'_2)$. By Lemma 2(a) and Weakening 1, $\Gamma \vdash s'[x := s] : \rho'_1$ and $\Gamma, z' : \rho'_2 \vdash t'[x := s] : \tau_1$. If $z' \notin \mathrm{FV}(t'[x := s])$ then we are done by (i) for type ρ which has been proved before. Otherwise, even $z' \in \mathrm{FV}(t')$ (since we may assume that $z' \notin \mathrm{FV}(s)$), hence $\rho' = \rho'_1 \to \rho'_2$. Consequently, $\Gamma, x : \rho \vdash \rho'_1 \to \rho'_2$, and by Lemma 3 $\rho = \cap\vec{\rho}$ with $\rho'_1 \to \rho'_2 \in \vec{\rho}$. Therefore $\Gamma \vdash s : \rho'_1 \to \rho'_2$. Hence, we are done by main induction hypothesis (i) if $\rho \notin \mathcal{P}$, and by (i) (proved beforehand) if $\rho = \rho'_1 \to \rho'_2$.

(i)(λ). Let $\lambda xr \in \mathsf{SN}$ be derived from $r \in \mathsf{SN}$. Show that $(\lambda xr)(s,z.t) \in \mathsf{SN}$. If $z \notin \mathrm{FV}(t)$ this follows by (β) from $r,s,t \in \mathsf{SN}$. Otherwise $\Gamma \vdash \lambda xr : \rho_1 \to \rho_2$. By Lemma 3, we get $\Gamma, x : \rho_1 \vdash r : \rho_2$. By the main induction hypothesis (ii) for ρ_1, $r[x := s] \in \mathsf{SN}$, and by the introductory remark $\Gamma \vdash r[x := s] : \rho_2$. By the main induction hypothesis (ii) for ρ_2, $t[z := r[x := s]] \in \mathsf{SN}$. Hence, we are done by rule (β). □

Theorem 4 (Soundness/Strong Normalization) *If* $\Gamma \vdash r : \rho$ *then* $r \in \mathsf{SN}$.

Proof By induction on typing derivations, using part (i) of the previous lemma. □

7 Open Problems

Does the method extend to an intersection type system for an untyped version of Gödel's system T? Is there also a strict intersection type system [10] for ΛJ which is complete and sound? Does it furnish an embedding into the simply-typed ΛJ like in [2] for usual λ-calculus? It seems that this question is much harder for ΛJ, namely, that reduction modulo some congruence relation has to be studied in the target system of the embedding. How is ΛJ related to calculi with continuations? The additional $z.t$ may be seen as the information on how to continue the calculation after applying r to s in $r(s,z.t)$.

[8] They also use Weakening 1, Weakening 2, Lemma 2(b), Lemma 3, Subject Reduction and Lemma 11 (in case of a substitution of the head variable in (ii)(π)).

References

[1] BARENDREGT, H. P. Lambda calculi with types. In vol. 2 of *Handbook of Logic in Computer Science*, S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, Eds., Oxford University Press, 1993, pp. 117–309.

[2] BUCCIARELLI, A., LORENZIS, S. D., PIPERNO, A., AND SALVO, I. Some Computational Properties of Intersection Types. In *14th Annual IEEE Symposium on Logic in Computer Science, Trento* (1999), IEEE Computer Society, pp. 109–118.

[3] GALLIER, J. Typing untyped λ-terms, or reducibility strikes again! *Annals of Pure and Applied Logic 91* (1998), 231–270.

[4] GOGUEN, H. Typed operational semantics. In *Proceedings of TLCA '95, Edinburgh* (1995), M. Dezani-Ciancaglini and G. Plotkin, Eds., vol. 902 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 186–200.

[5] JOACHIMSKI, F., AND MATTHES, R. Short proofs of normalization for the simply-typed lambda-calculus, permutative conversions and Gödel's T. Submitted, 1999. Available via the homepage.

[6] JOACHIMSKI, F., AND MATTHES, R. Standardization and confluence for a lambda calculus with generalized applications. To appear in the *Proceedings of RTA 2000, Norwich* (2000), *Lecture Notes in Computer Science*.

[7] KRIVINE, J.-L. *Lambda-calculus, types and models*. Masson, Paris and Ellis Horwood, Hemel Hempstead, 1993.

[8] MATTHES, R. *Extensions of System F by Iteration and Primitive Recursion on Monotone Inductive Types*. Doktorarbeit (Ph.D. thesis), University of Munich, 1998.

[9] TAIT, W. W. A realizability interpretation of the theory of species. In *Logic Colloquium Boston 1971/72* (1975), R. Parikh, Ed., vol. 453 of *Lecture Notes in Mathematics*, Springer Verlag, pp. 240–251.

[10] VAN BAKEL, S. Complete restrictions of the intersection type discipline. *Theoretical Computer Science 102*, 1 (1992), 135–163.

[11] VAN RAAMSDONK, F., AND SEVERI, P. On normalisation. Tech. Rep. CS-R9545, Centrum voor Wiskunde en Informatica (CWI), Amsterdam, 1995.

[12] VON PLATO, J. Natural deduction with general elimination rules. Submitted, 1998.

Ralph Matthes is with the Institut für Informatik, Ludwig-Maximilians-Universität München, 80538 München, Germany. E-mail: matthes@informatik.uni-muenchen.de