A Faithful and Quantitative Notion of Distant Reduction for Generalized Applications

José Espírito Santo[1][0000−0002−6348−5653], Delia Kesner[2,3][0000−0003−4254−3129], and Loïc Peyrot[2]

[1] Centro de Matemática, Universidade do Minho, Portugal
jes@math.uminho.pt
[2] Université de Paris, CNRS, IRIF, Paris, France
{kesner,lpeyrot}@irif.fr
[3] Institut Universitaire de France (IUF), France

Abstract. We introduce a call-by-name lambda-calculus λJ with generalized applications which integrates a notion of distant reduction that allows to unblock β-redexes without resorting to the permutative conversions of generalized applications. We show strong normalization of simply typed terms, and we then fully characterize strong normalization by means of a quantitative typing system. This characterization uses a non-trivial inductive definition of strong normalization –that we relate to others in the literature–, which is based on a weak-head normalizing strategy. Our calculus relates to explicit substitution calculi by means of a translation between the two formalisms which is faithful, in the sense that it preserves strong normalization. We show that our calculus λJ and the well-know calculus ΛJ determine equivalent notions of strong normalization. As a consequence, ΛJ inherits a faithful translation into explicit substitutions, and its strong normalization can be characterized by the quantitative typing system designed for λJ, despite the fact that quantitative subject reduction fails for permutative conversions.

1 Introduction

(Pure) functional programming can be understood by means of a universal model of computation known as the λ-calculus, which is in tight correspondence, by means of the so-called Curry-Howard isomorphism, with propositional intuitionistic logic in Gentzen's natural deduction style. The Curry-Howard isomorphism emphasizes the fact that proof systems on one hand, and programming languages on the other, are two mathematical and computational facets of the same object. The λ-calculus with *generalized applications* (ΛJ), introduced by Joachimski and Matthes [8], is an extension of the λ-calculus which can be seen as the Curry-Howard counterpart of van Plato's natural deduction with generalized elimination rules [11].

A generalized application in ΛJ is written $t(u, y.r)$. It intuitively means that t is applied to u in the context of the substitution $\{\_/y\}r$. The conversion of the β-redex $(\lambda x.t)(u, y.r)$ then produces two (nested) substitutions $\{\{u/x\}t/y\}r$. But some β-redexes can be *blocked* by the syntax, *e.g.* in the term $t(u, y.r)(u', y'.r')$, where the (potential) application of $r = \lambda x.s$ to u' remains hidden. An iterated generalized application $t(u, y.r)(u', y'.r')$ may be rearranged as $t(u, y.r(u', y'.r'))$ by a permutative conversion called π. Rule π is then an unblocker of stuck β-redexes: the contractum $t(u, y.(\lambda x.s)(u', y'.r'))$ unveils the desired application of r to u'. Rule π, together with rule β, allows natural deduction proofs to be brought to a "fully normal" form [11] enjoying the subformula property. Computationally, ΛJ defines a call-by-name operational semantics; a call-by-value variant has been proposed in [5], but this is out of the scope of this paper.

Strong normalization w.r.t. the two rules β and π has been characterized by typability with (idempotent) intersection types by Matthes [10]: a term is typable if and only if it is strongly normalizing. However, this characterization is just *qualitative.* A different flavor of intersection types, called *non-idempotent*, offers a more powerful *quantitative* characterization of strong normalization, in the sense that the length of the longest reduction sequence to normal form starting at a typable term t is bound by the size of its type derivation. However, quantitative types were never used in the framework of generalized applications, and it is our purpose to propose and study one such typing system.

Quantitative types allow for simple combinatorial proofs of strong normalization, without any need to use reducibility or computability arguments. More remarkably, they also provide a refined tool to understand permutative rules. For instance, in ΛJ, rule π is not quantitatively sound (*i.e.* π does not enjoy quantitative subject reduction), although π becomes valid in an idempotent framework. Hence, a good question is: how can we unblock redexes to reach normal forms in a quantitative model of computation based on generalized applications?

Our solution is to adopt the paradigm of *distant* reduction [2] coming from explicit substitution (ES) calculi, which extends the key concept of β-redex, so that we may find the λ-abstraction hidden under a sequence of nested generalized applications. This is essentially similar to adopting a different permutation rule, converting $t(u, y.\lambda z.s)$ to $\lambda z.t(u, y.s)$. However, the permutation rule is mostly a way to overcome syntactical limitations, while distant β is a way to put emphasis on the computational behavior of the calculus: it is at the β-step that resources are consumed, not during the permutations.

The syntax of the ΛJ-calculus will thus be equipped with an operational call-by-name semantics given by distant β, but without π. The resulting calculus is called λJ. As a major contribution, we prove a characterization of strong normalization in terms of typability in our quantitative system. In such proof, the soundness result (typability implies strong normalization) is obtained by combinatorial arguments, with the size of typing derivations decreasing at each step. For the completeness result (strong normalization implies typability) we need an inductive characterization of the terms that are strongly normalizing for distant β: this is a non-trivial technical contribution of the paper.

As mentioned above, we draw inspiration for our distant β rule from calculi with explicit substitutions, having in mind the usual translation of $t(u, y.r)$ to the explicit substitution $[tu/y]r$ (a let-binding of tu over y in r). As such, we expect the dynamic behavior of our calculus to be *faithful* to explicit substitutions. Such translation, however, does not in general preserve strong normalization. Indeed, in a β-redex $(\lambda x.t)(u, y.r)$, the interaction of $\lambda x.t$ with the argument u is materialized by the internal substitution in the contractum term $\{\{u/x\}t/y\}r$, as mentioned before. But such interaction is elusive: if the external substitution is vacuous (that is, if y is not free in r), β-reduction will simply throw away the λ-abstraction $\lambda x.t$ and its argument u, whereas $(\lambda x.t)u$ may reduce in the context of the explicit substitution $[(\lambda x.t)u/y]r$. The different interaction between the abstraction and its argument in the two mentioned models of computation has important consequences. For instance, let $\delta^\circ := \lambda x.x(x, w.w)$ be the encoding of $\delta = \lambda x.xx$ as a ΛJ-term. Then, if $y \notin r$ and r is normal, the only thing the term $\delta^\circ(\delta^\circ, y.r)$ can do is to reduce to r, whereas $\delta\delta$ may reduce forever in the context of the vacuous explicit substitution $[\delta\delta/y]r$.

That is why we propose a new, type-preserving, encoding of terms with generalized applications into terms with explicit substitutions. Using this new encoding and quantitative types, we show that strong normalization of the source term with generalized applications is equivalent to the strong normalization of the target term with explicit substitutions.

As a final contribution, we compare λJ-strong normalization to that of other calculi, including the original ΛJ. We extract new results for the latter, as a faithful translation to ES, and a new normalizing strategy. Moreover, we obtain a quantitative characterization of ΛJ-strong normalization, where the bound for reduction given by the size of type derivations only holds for β (and not for π).

Plan of the paper. Sec. 2 presents our calculus with distant β. Sec. 3 provides an inductive characterization of strongly normalizing terms. Sec. 4 is about non-idempotent intersection types. Sec. 5 shows the faithful translation to ES. Sec. 6 contains the comparisons with other calculi. Sec. 7 concludes. Full proofs are available in [6].

2 A Calculus with Generalized Applications

In this section we define our calculus λJ with generalized applications and give some introductory observations on strong normalization in that system.

2.1 Syntax and Semantics

We start with some general notations. Given a reduction relation $\to_{\mathcal{R}}$, we write $\to^*_{\mathcal{R}}$ (resp. $\to^+_{\mathcal{R}}$) for the reflexive-transitive (resp. transitive) closure of $\to_{\mathcal{R}}$. A term t is said to be in **$\mathcal{R}$-normal form** (written $\mathcal{R}$-nf) iff there is no t' such that $t \to_{\mathcal{R}} t'$. A term t is said to be **$\mathcal{R}$-strongly normalizing** (written $t \in \mathcal{SN}(\mathcal{R})$) iff there is no infinite $\mathcal{R}$-sequence starting at t. $\mathcal{R}$ is strongly normalizing iff every term is $\mathcal{R}$-strongly normalizing. When $\mathcal{R}$ is finitely branching, $\|t\|_{\mathcal{R}}$ denotes the maximal length of an $\mathcal{R}$-reduction sequence to $\mathcal{R}$-nf starting at t, for every $t \in \mathcal{SN}(\mathcal{R})$.

The set of terms generated by the following grammar is denoted by $\mathtt{T}_J$.

$$(\textbf{Terms})\ t,u,r,s ::= x \mid \lambda x.t \mid t(u,x.r)$$

The term $t(u,x.r)$ is called a generalized application, and the part $x.r$ is sometimes referred as the *continuation* of that application. Free variables of terms are defined as usual, notably $\mathrm{fv}(t(u,x.r)) := \mathrm{fv}(t) \cup \mathrm{fv}(u) \cup \mathrm{fv}(r) \setminus \{x\}$. We also work modulo α-conversion, denoted $=_\alpha$, so that bound variables can be systematically renamed. We use $\mathtt{I}$ to denote the identity function $\lambda z.z$.

We introduce contexts (terms with one occurrence of the hole $\Diamond$) and the special distant contexts:

$$\begin{array}{ll}(\textbf{Contexts}) & \mathtt{C} ::= \Diamond \mid \lambda x.\mathtt{C} \mid \mathtt{C}(u,x.r) \mid t(\mathtt{C},x.r) \mid t(u,x.\mathtt{C}) \\ (\textbf{Distant Contexts}) & \mathtt{D} ::= \Diamond \mid t(u,x.\mathtt{D})\end{array}$$

The term $\mathtt{C}\langle t\rangle$ denotes $\mathtt{C}$ where $\Diamond$ is replaced by t, so that capture of variables may eventually occur. Given a rewriting rule $\mathcal{R} \subseteq \mathtt{T}_J \times \mathtt{T}_J$, $\to_\mathcal{R}$ denotes the reduction relation generated by the closure of $\mathcal{R}$ under all contexts.

We say that t has an **abstraction shape** iff $t = \mathtt{D}\langle \lambda x.u\rangle$. The substitution operation is capture-avoiding and defined as usual, in particular $\{u/x\}(t(s,y.r)) := (\{u/x\}t)(\{u/x\}s, y.\{u/x\}r)$.

2.2 Towards a Call-by-Name Operational Semantics

The $\mathtt{T}_J$-syntax can be equipped with different rewriting rules, as discussed in the introduction. We use the generic notation $\mathtt{T}_J[\mathcal{R}]$ to denote the calculus given by the syntax $\mathtt{T}_J$ equipped with the reduction relation $\to_\mathcal{R}$.

Now, if we consider $t_0 := t(u', y'.\lambda x.s)(u,y.r)$ in the calculus $\mathtt{T}_J[\beta]$, where

$$(\lambda x.s)(u,y.r) \mapsto_\beta \{\{u/x\}s/y\}r$$

we can see that the term t_0 is stuck since the subterm $\lambda x.s$ is not close to u. This is when the following rule π, plays the role of an unblocker of β-redexes:

$$t(u,y.r)(u',y'.r') \mapsto_\pi t(u,y.r(u',y'.r'))$$

Indeed, $t_0 \to_\pi t(u', y'.(\lambda x.s)(u,y.r)) \to_\beta t(u',y'.\{\{u/x\}s/y\}r)$. More generally, given $t_1 := \mathtt{D}\langle \lambda x.s\rangle(u,y.r)$, with $\mathtt{D} \neq \Diamond$, a sequence of π-steps reduces the term t_1 above to $\mathtt{D}\langle(\lambda x.s)(u,y.r)\rangle$. A further β-step produces $\mathtt{D}\langle\{\{u/x\}s/y\}r\rangle$. So, the original ΛJ-calculus [8], which is exactly $\mathtt{T}_J[\beta,\pi]$, has a derived notion of *distant* β rule, based on π, which can be specified by the following rule:

$$\mathtt{D}\langle \lambda x.s\rangle(u,y.r) \mapsto \mathtt{D}\langle\{\{u/x\}s/y\}r\rangle \tag{1}$$

However, π-reduction is not only about unblocking redexes, as witnessed by $\mathtt{D}\langle x\rangle(u,y.r) \to_\pi^* \mathtt{D}\langle x(u,y.r)\rangle$. So it is reasonable to keep terms of the form $\mathtt{D}\langle x\rangle(u,y.r)$ without reducing them further, as those π-steps do not contribute to unblock more β-redexes. The absence of terms of the form $\mathtt{D}\langle \lambda x.s\rangle(u,y.r)$

gives already a reasonable notion of normal form which, in particular, already enjoy the subformula property, as will be seen in Sec. 2.3.

Still, we will not reduce as in (1) because such rule, as well as π itself, does not admit a quantitative semantics (*c.f.* Sec. 4.3). We then choose to unblock β-redexes with the following rule $\mathtt{p_2}$ instead[4]:

$$t(u', y'.\lambda x.s) \mapsto_{\mathtt{p_2}} \lambda x.t(u', y'.s)$$

so that t_1 given above reduces in several $\mathtt{p_2}$-steps to $(\lambda x.\mathtt{D}\langle s\rangle)(u, y.r)$, which can now be further reduced with β since it is no longer stuck. If we reduce it, we obtain $\{\{u/x\}\mathtt{D}\langle s\rangle/y\}r$; and since free variables in u cannot be captured by $\mathtt{D}$, this is equal to $\{\mathtt{D}\langle\{u/x\}s\rangle/y\}r$. We thus obtain our distant rule:

Definition 1. *We write λJ for our new calculus $\mathtt{T}_J[\mathtt{d}\beta]$, where the distant β-rule is defined as follows:*

$$\mathtt{D}\langle\lambda x.t\rangle(u, y.r) \mapsto_{\mathtt{d}\beta} \{\mathtt{D}\langle\{u/x\}t\rangle/y\}r$$

A reduction step $t_1 \to_{\mathtt{d}\beta} t_2$ is said to be **erasing** iff the reduced $\mathtt{d}\beta$-redex in t_1 is of the form $\mathtt{D}\langle\lambda x.t\rangle(u, y.r)$ with $x \notin \mathrm{fv}(t)$ or $y \notin \mathrm{fv}(r)$.

It is obvious that $\to_{\mathtt{d}\beta} \subset \to^{+}_{\beta,\mathtt{p_2}}$. Some other variants of the $\mathtt{p_2}$-rule are possible, like $\mathtt{D}\langle\lambda x.t\rangle(u, y.r) \mapsto (\lambda x.\mathtt{D}\langle t\rangle)(u, y.r)$ or $\mathtt{D}\langle\lambda x.t\rangle \mapsto_{\mathtt{p_2}} \lambda x.\mathtt{D}\langle t\rangle$, in both cases for $\mathtt{D} \neq \Diamond$, but we do not develop them. However, while most of the paper is about λJ, brief comparisons with the calculi ΛJ and $\mathtt{T}_J[\beta, \mathtt{p_2}]$ are considered in Sec.6.

2.3 Some (Un)typed Properties of λJ

The following grammar characterizes $\mathtt{d}\beta$-normal forms:

$$\mathtt{m} ::= x \mid \lambda x.\mathtt{m} \mid \mathtt{m_{var}}(\mathtt{m}, x.\mathtt{m}) \qquad \mathtt{m_{var}} ::= x \mid \mathtt{m_{var}}(\mathtt{m}, x.\mathtt{m_{var}})$$

Lemma 1. *Let t be a term. Then $t \in \mathtt{m}$ iff t is a $\mathtt{d}\beta$-nf.*

We already saw that, once β is generalized to $\mathtt{d}\beta$, π is not needed anymore to unblock β-redexes; the next Lemma says that π preserves $\mathtt{d}\beta$-nfs, so it does not bring anything new to $\mathtt{d}\beta$-nfs either. The proof uses Lem. 1, and it proceeds by simultaneous induction on $\mathtt{m}$ and $\mathtt{m_{var}}$.

Lemma 2. *If t is a $\mathtt{d}\beta$-nf, and $t \to_\pi t'$, then t' is a $\mathtt{d}\beta$-nf.*

Let us discuss now some properties related to (simple) typability for generalized applications [8], a system that we call ST. Recall the following typing rules, where $\sigma, \rho, \tau ::= a \mid \sigma \to \rho$, and a belongs to a set of base type variables:

$$\frac{}{\Gamma, x : \sigma \vdash x : \sigma} \qquad \frac{\Gamma, x : \sigma \vdash t : \rho}{\Gamma \vdash \lambda x.t : \sigma \to \rho} \qquad \frac{\Gamma \vdash t : \rho \to \tau \quad \Gamma \vdash u : \rho \quad \Gamma, y : \tau \vdash r : \sigma}{\Gamma \vdash t(u, y.r) : \sigma}$$

We write $\Gamma \Vdash_{ST} t : \sigma$ if there is a type derivation in system ST ending in $\Gamma \vdash t : \sigma$. In the following result, we refer to simple types as formulas.

[4] Rule $\mathtt{p_2}$ is used in [7, 3] along with two other permutation rules $\mathtt{p_1}$ and $\mathtt{p_3}$ to reduce $\mathtt{T}_J$-terms to a fragment isomorphic to natural deduction.

Lemma 3 (Subformula Property). *If $\Phi = \Gamma \Vdash_{ST} \mathtt{m} : \tau$ then every formula in the derivation Φ is a subformula of τ or a subformula of some formula in Γ.*

Proof. The lemma is proved together with another statement: If $\Psi = \Gamma \Vdash_{ST} \mathtt{m}_{\mathtt{var}} : \tau$ then every formula in Ψ is a subformula of some formula in Γ. The proof is by simultaneous induction of Φ and Ψ.

We close this section with the following:

Theorem 1. *If t is simply typable, i.e. $\Gamma \Vdash_{ST} t : \sigma$, then $t \in \mathcal{SN}(\mathtt{d}\beta)$.*

The proof is by a map into the λ-calculus which produces a simulation when the λ-calculus is equipped with the following σ-rules [13]:

$$(\lambda x.M)NN' \mapsto_{\sigma_1} (\lambda x.MN')N \qquad (\lambda x.\lambda y.M)N \mapsto_{\sigma_2} \lambda y.(\lambda x.M)N$$

3 Inductive Characterization of Strong Normalization

In this section we give an inductive characterization of strong normalization (ISN) for λJ and prove it correct. This characterization will be useful to show completeness of the type system that we are going to present in Sec. 4.1, as well as to compare strong normalization of λJ to the ones of $\mathtt{T}_\lambda[\beta, \mathtt{p_2}]$ and ΛJ.

3.1 ISN in the λ-Calculus Through Weak-Head Contexts

As an introduction, we first look at the case of the ISN for the λ-calculus ($\mathcal{ISN}(\beta)$), on which our forthcoming definition of $\mathcal{ISN}(\mathtt{d}\beta)$ elaborates. A usual way to define $\mathcal{ISN}(\beta)$ is by the following rules [12], where the general notation $t\boldsymbol{r}$ abbreviates $(\dots(tr_1)\dots)r_n$ for some $n \geq 0$.

$$\frac{r_1, \dots, r_n \in \mathcal{ISN}(\beta)}{x\boldsymbol{r} \in \mathcal{ISN}(\beta)} \qquad \frac{t \in \mathcal{ISN}(\beta)}{\lambda x.t \in \mathcal{ISN}(\beta)} \qquad \frac{\{u/x\}t\boldsymbol{r}, u \in \mathcal{ISN}(\beta)}{(\lambda x.t)u\boldsymbol{r} \in \mathcal{ISN}(\beta)}$$

One shows that $t \in \mathcal{SN}(\beta)$ if and only if $t \in \mathcal{ISN}(\beta)$.

The reduction strategy underlying the definition of $\mathcal{ISN}(\beta)$ is the following one: reduce terms to weak-head normal form, and then iterate reduction inside the components of the weak-head normal form, without any need to come back to the head of the term. **Weak-head normal terms** are of two kinds: (**neutral terms**) $\mathtt{n} ::= x \mid \mathtt{n}t$ and (**answers**) $\mathtt{a} ::= \lambda x.t$. Neutral terms cannot produce any head β-redex. On the contrary, answers can create a β-redex when given at least one argument. In the case of the λ-calculus, these are only abstractions. If the term is not a weak-head term, a redex can be located with a *weak-head context* $\mathtt{W} ::= \Diamond \mid \mathtt{W}t$. These concepts allow a different definition of $\mathcal{ISN}(\beta)$.

$$\frac{}{x \in \mathcal{ISN}(\beta)} \qquad \frac{\mathtt{n}, t \in \mathcal{ISN}(\beta)}{\mathtt{n}t \in \mathcal{ISN}(\beta)} \qquad \frac{t \in \mathcal{ISN}(\beta)}{\lambda x.t \in \mathcal{ISN}(\beta)} \qquad \frac{\mathtt{W}\langle\{u/x\}t\rangle, u \in \mathcal{ISN}(\beta)}{\mathtt{W}\langle(\lambda x.t)u\rangle \in \mathcal{ISN}(\beta)}$$

Weak-head contexts are an alternative to the meta-syntactic notation $\boldsymbol{r}$ of vectors of arguments. Notice that there is one rule for each kind of neutral term, one rule for answers and one rule for terms which are not weak-head normal forms.

3.2 ISN for $\mathtt{d}\beta$

We define $\mathcal{ISN}(\mathtt{d}\beta)$ with the same methodology as before. Hence, we first have to define neutral terms, answers and weak-head contexts.

Definition 2. *We consider the following grammars:*

$$\begin{array}{rl} (\textbf{Neutral terms})\ \mathtt{n} & ::= x \mid \mathtt{n}(u, x.\mathtt{n}) \\ (\textbf{Answers})\ \mathtt{a} & ::= \lambda x.t \mid \mathtt{n}(u, x.\mathtt{a}) \\ (\textbf{Neutral distant contexts})\ \mathtt{D_n} & ::= \Diamond \mid \mathtt{n}(u, x.\mathtt{D_n}) \\ (\textbf{Weak-head contexts})\ \mathtt{W} & ::= \Diamond \mid \mathtt{W}(u, x.r) \mid \mathtt{n}(u, x.\mathtt{W}) \end{array}$$

Notice that $\mathtt{n}$ *and* $\mathtt{a}$ *are disjoint and stable by* $\mathtt{d}\beta$*-reduction. Also* $\mathtt{D_n} \subsetneq \mathtt{W}$*.*

Example 1 (Decomposition). Let $t = x_1(x_2, y_1.I(I, z.I))(x_3, y.II)$. Then, there are two decompositions of t in terms of a redex r and a weak-head context $\mathtt{W}$: either $\mathtt{W} = \Diamond$ and $r = t$, or $\mathtt{W} = x_1(x_2, y_1.\Diamond)(x_3, y.II)$ and $r = I(I, z.I)$. In both cases $t = \mathtt{W}\langle r\rangle$. We will rule out the first possibility by defining next a restriction of the β-rule, securing uniqueness of such kind of decomposition in all cases.

The strategy underlying our definition of $\mathcal{ISN}(\mathtt{d}\beta)$ will be the **weak-head strategy** $\to_{\mathrm{wh}}$, defined as the closure under $\mathtt{W}$ of the following restricted β-rule:

$$\mathtt{D_n}\langle\lambda x.t\rangle(u, y.r) \mapsto \{\mathtt{D_n}\langle\{u/y\}t\rangle/y\}r.$$

The restriction of $\mathtt{D}$ to a neutral distant context $\mathtt{D_n}$ is what allows determinism of our forthcoming Def. 3.

Lemma 4. *The reduction* $\to_{\mathrm{wh}}$ *is deterministic.*

As in the case of the λ-calculus, weak-head normal forms are either neutral terms or answers. This time, answers are not only abstractions, but also abstractions under a (neutral) distant context. Because of distance, these terms can also create a $\mathtt{d}\beta$-redex when applied to an argument, as seen in the next example.

Example 2. Consider again term t of Ex. 1. If the third form in the grammar of $\mathtt{W}$ was disallowed, then it would not be possible to write t as $\mathtt{W}\langle r\rangle$, with r a restricted redex. In that case, the reduction strategy associated with $\mathcal{ISN}(\mathtt{d}\beta)$ would consider t as a weak-head normal form, and start reducing the subterms of t, including $I(I, z.I)$. Now, the latter would eventually reach I and suddenly the whole term $t' = x_1(x_2, y_1.I)(x_3, y.r')$ would be a weak-head redex again: the typical separation between an initial weak-head reduction phase and a later internal reduction phase, as it is the case in the λ-calculus, would be lost in our framework. This is a subtle point due to the *distant* character of rule $\mathtt{d}\beta$ which explains the complexity of Def. 2.

Lemma 5. *Let* $t \in \mathtt{T}_J$*. Then* t *is in* wh*-normal form iff* $t \in \mathtt{n} \cup \mathtt{a}$*.*

Our inductive definition of strong normalization follows.

Definition 3 (Inductive Strong Normalization). *We consider the following inductive predicate:*

$$\frac{}{x \in \mathcal{ISN}(\mathtt{d}\beta)}(\mathtt{snvar}) \qquad \frac{\mathtt{n}, u, r \in \mathcal{ISN}(\mathtt{d}\beta) \qquad r \in \text{wh-}nf}{\mathtt{n}(u, x.r) \in \mathcal{ISN}(\mathtt{d}\beta)}(\mathtt{snapp})$$

$$\frac{t \in \mathcal{ISN}(\mathtt{d}\beta)}{\lambda x.t \in \mathcal{ISN}(\mathtt{d}\beta)}(\mathtt{snabs}) \qquad \frac{\mathtt{W}\langle\{\mathtt{D_n}\langle\{u/x\}t\rangle/y\}r\rangle, \mathtt{D_n}\langle t\rangle, u \in \mathcal{ISN}(\mathtt{d}\beta)}{\mathtt{W}\langle\mathtt{D_n}\langle\lambda x.t\rangle(u, y.r)\rangle \in \mathcal{ISN}(\mathtt{d}\beta)}(\mathtt{snbeta})$$

Notice that every term can be written according to the conclusions of the previous rules, so that the grammar $t, u, r ::= x \mid \lambda x.t \mid \mathtt{n}(t, x.r) \mid \mathtt{W}\langle\mathtt{D_n}\langle\lambda x.t\rangle(u, y.s)\rangle$, with $r \in$ wh-nf, also defines the syntax $\mathtt{T}_J$. Moreover, at most one rule in the previous definition applies to each term, *i.e.* the rules are deterministic. An equivalent, but non-deterministic definition, can be given by removing the side condition "$r \in$ wh-nf" in rule ($\mathtt{snapp}$). Indeed, this (weaker) rule would overlap with rule ($\mathtt{snbeta}$) for terms in which the weak-head context lies in the last continuation, as for instance in $x(u, y.y)(u', y'.\mathtt{II})$. Notice the difference with the λ-calculus: the head of a term with generalized applications can be either on the left of the term (as in the λ-calculus), or recursively on the left in a continuation. We conclude with the following result.

Theorem 2. $\mathcal{SN}(\mathtt{d}\beta) = \mathcal{ISN}(\mathtt{d}\beta)$.

4 Quantitative Types Characterize Strong Normalization

We proved that simply typable terms are strongly normalizing in Sec. 2.3. In this section we use non-idempotent intersection types to fully characterize strong normalization, so that strongly normalizing terms are also typable. First we introduce the typing system, next we prove the characterization and finally we study the quantitative behavior of π and give in particular an example of failure.

4.1 The Typing System

We now define our quantitative type system $\cap J$ for $\mathtt{T}_J$-terms and we show that strong normalization in λJ exactly corresponds to $\cap J$ typability.

Given a countable infinite set BTV of base type variables $a, b, c, \ldots$, we define the following sets of types:

$$\begin{array}{c}(\textbf{types})\ \sigma, \tau, \rho ::= a \in BTV \mid \mathcal{M} \to \sigma \\ (\textbf{multiset types})\ \mathcal{M}, \mathcal{N} ::= [\sigma_i]_{i \in I} \text{ where } I \text{ is a finite set}\end{array}$$

The empty multiset is denoted $[\,]$. We use $|\mathcal{M}|$ to denote the size of the multiset, thus if $\mathcal{M} = [\sigma_i]_{i \in I}$ then $|\mathcal{M}| = |I|$. We introduce a **choice operator** on multiset types: if $\mathcal{M} \neq [\,]$, then $\#(\mathcal{M}) = \mathcal{M}$, otherwise $\#([\,]) = [\sigma]$, where σ is an arbitrary type. This operator is used to guarantee that there is always a typing witness for all the subterms of typed terms.

Typing environments (or just **environments**), written Γ, Δ, Λ, are functions from variables to multiset types assigning the empty multiset to all but a finite set of variables. The domain of Γ is given by $\mathrm{dom}(\Gamma) := \{x \mid \Gamma(x) \neq [\,]\}$. The **union of environments**, written $\Gamma \wedge \Delta$, is defined by $(\Gamma \wedge \Delta)(x) := \Gamma(x) \sqcup \Delta(x)$, where $\sqcup$ denotes multiset union. This notion is extended to several environments as expected, so that $\wedge_{i \in I} \Gamma_i$ denotes a finite union of environments ($\wedge_{i \in I} \Gamma_i$ is to be understood as the empty environment when $I = \emptyset$). We write $\Gamma \setminus\!\!\setminus x$ for the environment such that $(\Gamma \setminus\!\!\setminus x)(y) = \Gamma(y)$ if $y \neq x$ and $(\Gamma \setminus\!\!\setminus x)(x) = [\,]$. We write $\Gamma; \Delta$ for $\Gamma \wedge \Delta$ when $\mathrm{dom}(\Gamma) \cap \mathrm{dom}(\Delta) = \emptyset$. A sequent has the form $\Gamma \vdash t : \sigma$, where Γ is an environment, t is a term, and σ is a type.

The type system $\cap J$ is given by the following typing rules.

$$\frac{}{x : [\sigma] \vdash x : \sigma}\ (\texttt{var}) \qquad \frac{\Gamma; x : \mathcal{M} \vdash t : \sigma}{\Gamma \vdash \lambda x.t : \mathcal{M} \to \sigma}\ (\texttt{abs}) \qquad \frac{(\Gamma_i \vdash t : \sigma_i)_{i \in I} \quad I \neq \emptyset}{\wedge_{i \in I} \Gamma_i \vdash t : [\sigma_i]_{i \in I}}\ (\texttt{many})$$

$$\frac{\Gamma \vdash t : \#([\mathcal{M}_i \to \tau_i]_{i \in I}) \qquad \Delta \vdash u : \#(\sqcup_{i \in I} \mathcal{M}_i) \qquad \Lambda; x : [\tau_i]_{i \in I} \vdash r : \sigma}{\Gamma \wedge \Delta \wedge \Lambda \vdash t(u, x.r) : \sigma}\ (\texttt{app})$$

The use of the choice operator in rule $(\texttt{app})$ is subtle. If I is empty, then the multiset $[\mathcal{M}_i \to \tau_i]_{i \in I}$ typing t as well as the multiset $\sqcup_{i \in I} \mathcal{M}_i$ typing u are both empty, so that the choice operator must be used to type both terms. If I is not empty, then the multiset typing t is non-empty as well. However, the multiset typing u may or not be empty, *e.g.* if $[[\,] \to \alpha]$ types t.

System $\cap J$ lacks weakening: it is *relevant*.

Lemma 6 (Relevance). *If* $\Gamma \Vdash t : \sigma$, *then* $\mathrm{fv}(t) = \mathrm{dom}(\Gamma)$.

Notice that the typing rules (and the choice operator) force all the subterms of a typed term to be also typed. Moreover, if $I = \emptyset$ in rule $(\texttt{app})$, then the types of t and u are not necessarily related. Indeed, let $\delta^\circ := \lambda y.y(y, w.w)$ in $t_0 := \delta^\circ(\delta^\circ, x.z)$. Then t_0 is $\mathtt{d}\beta$-strongly-normalizing so it must be typed in system $\cap J$. However, since the set I of $x : [\tau_i]_{i \in I}$ in the typing of $r = z$ is necessarily empty (*c.f.* Lem. 6), then the unrelated types $\#([\mathcal{M}_i \to \tau_i]_{i \in I})$ and $\#(\sqcup_{i \in I} \mathcal{M}_i)$ of the two occurrences of δ° witness to the fact that these subterms will never interact during the reduction of t_0. Indeed, the term t_0 can be typed as follows, where $\rho_i := [[\sigma_i] \to \sigma_i, \sigma_i] \to \sigma_i$ and $\tau_i := [\sigma_i] \to \sigma_i$, for $i = 1, 2$:

$$\frac{\dfrac{\emptyset \vdash \delta^\circ : \rho_1}{\emptyset \vdash \delta^\circ : [\rho_1]}\ (\texttt{many}) \qquad \dfrac{\emptyset \vdash \delta^\circ : \rho_2}{\emptyset \vdash \delta^\circ : [\rho_2]}\ (\texttt{many}) \qquad \dfrac{}{z : [\tau]; x : [\,] \vdash z : \tau}\ (\texttt{var})}{z : [\tau] \vdash \delta^\circ(\delta^\circ, x.z) : \tau}\ (\texttt{app})$$

where δ° is typed with ρ_i as follows:

$$\dfrac{\dfrac{\dfrac{}{y : [\tau_i] \vdash y : \tau_i}\ (\texttt{var})}{y : [\tau_i] \vdash y : [\tau_i]}\ (\texttt{many}) \qquad \dfrac{\dfrac{}{y : [\sigma_i] \vdash y : \sigma_i}\ (\texttt{var})}{y : [\sigma_i] \vdash y : [\sigma_i]}\ (\texttt{many}) \qquad \dfrac{}{w : [\sigma_i] \vdash w : \sigma_i}\ (\texttt{var})}{\dfrac{y : [[\sigma_i] \to \sigma_i, \sigma_i] \vdash y(y, w.w) : \sigma_i}{\emptyset \vdash \lambda y.y(y, w.w) : [[\sigma_i] \to \sigma_i, \sigma_i] \to \sigma_i}\ (\texttt{abs})}\ (\texttt{app})$$

We write $\Gamma \Vdash_{\cap J} t : \sigma$ or simply $\Gamma \Vdash t : \sigma$ if there is a derivation in system $\cap J$ ending in $\Gamma \vdash t : \sigma$. For $n \geq 1$, we write $\Gamma \Vdash^{n}_{\cap J} t : \sigma$ or simply $\Gamma \Vdash^{n} t : \sigma$ if there is a derivation in system $\cap J$ ending in $\Gamma \vdash t : \sigma$ and containing n occurrences of rules in the set $\{(\mathtt{var}), (\mathtt{abs}), (\mathtt{app})\}$.

4.2 The Characterization of dβ-Strong Normalization

The soundness Lem. 9 is based on Lem. 8, based in turn on Lem. 7.

Lemma 7 (Substitution Lemma). *Let $t, u \in \mathtt{T}_J$ with $x \in \mathrm{fv}(t)$. If both $\Gamma; x : \mathcal{M} \Vdash^{n} t : \sigma$ and $\Delta \Vdash^{m} u : \mathcal{M}$ hold, then $\Gamma \wedge \Delta \Vdash^{k} \{u/x\}t : \sigma$ where $k = n + m - |\mathcal{M}|$.*

Lemma 8 (Non-Erasing Subject Reduction). *Let $\Gamma \Vdash^{n_1}_{\cap J} t_1 : \sigma$. If $t_1 \to_{\mathtt{d}\beta} t_2$ is a non-erasing step, then $\Gamma \Vdash^{n_2}_{\cap J} t_2 : \sigma$ with $n_1 > n_2$.*

Lemma 9 (Soundness for λJ). *If t is $\cap J$-typable, then $t \in \mathcal{SN}(\mathtt{d}\beta)$.*

The completeness Lem. 13 is based on Lem. 10 and Lem. 12, this last based in turn on Lem. 11.

Lemma 10 (Typing Normal Forms).

1. *For all $t \in \mathtt{m}$, there exists Γ, σ such that $\Gamma \Vdash_{\cap J} t : \sigma$.*
2. *For all $t \in \mathtt{m}_{\mathtt{var}}$, for all σ, there exists Γ such that $\Gamma \Vdash_{\cap J} t : \sigma$.*

Lemma 11 (Anti-Substitution). *If $\Gamma \Vdash \{u/x\}t : \sigma$ where $x \in \mathrm{fv}(t)$, then there exist Γ_t, Γ_u and $\mathcal{M} \neq [\,]$ such that $\Gamma_t; x : \mathcal{M} \Vdash t : \sigma$, $\Gamma_u \Vdash u : \mathcal{M}$ and $\Gamma = \Gamma_t \wedge \Gamma_u$.*

Lemma 12 (Non-Erasing Subject Expansion). *If $\Gamma \Vdash_{\cap J} t_2 : \sigma$ and $t_1 \to_{\mathtt{d}\beta} t_2$ is a non-erasing step, then $\Gamma \Vdash_{\cap J} t_1 : \sigma$.*

Lemma 13 (Completeness for λJ). *If $t \in \mathcal{SN}(\mathtt{d}\beta)$, then t is $\cap J$-typable.*

We finally obtain:

Theorem 3 (Characterization). *System $\cap J$ characterizes strong normalization, i.e. t is $\cap J$-typable if and only if t is $\to_{\mathtt{d}\beta}$-normalizing. Moreover, if $\Gamma \Vdash^{n} t : \sigma$ then the number of reduction steps in any reduction sequence from t to normal form is bounded by n.*

Proof. Soundness holds by Lem. 9, while completeness holds by Lem. 13. The bound is given by Thm. 9 in the long version [6].

4.3 Why π Is Not Quantitative

In the introduction we discussed that π is rejected by the quantitative type systems $\cap J$ for CBN. This happens in the critical case when $x \notin \mathrm{fv}(r)$ and $y \in \mathrm{fv}(r')$ in $t_0 = t(u, x.r)(u', y.r') \to_\pi t(u, x.r(u', y.r')) = t_1$. Let us see a concrete example.

Example 3. We take $t_1 = x(y, a.z)(w, b.b(b, c.c)) \to_\pi x(y, a.z(w, b.b(b, c.c))) = t_2$. Let $\rho_1 = [\sigma] \to \tau$ and $\rho_2 = [\sigma] \to [\tau] \to \tau$. For each $i \in \{1, 2\}$ let $\Delta_i = x : [\sigma_1]; y : [\sigma_2]; z : [\rho_i]$. Consider

$$\Psi = \dfrac{b : [[\tau] \to \tau] \Vdash b : [[\tau] \to \tau] \qquad b : [\tau] \Vdash b : [\tau] \qquad \overline{c : [\tau] \vdash c : \tau}}{b : [[\tau] \to \tau, \tau] \vdash b(b, c.c) : \tau}$$

and the derivation Φ_i for $i \in \{1, 2\}$:

$$\Phi_i = \dfrac{x : [\sigma_1] \Vdash x : [\sigma_1] \qquad y : [\sigma_2] \Vdash y : [\sigma_2] \qquad \overline{z : [\rho_i] \vdash z : \rho_i}}{\Delta_i \vdash x(y, a.z) : \rho_i}$$

Then, for the term t_1, we have the following derivation:

$$\dfrac{\dfrac{\Phi_1 \qquad \Phi_2}{\Delta_1 \wedge \Delta_2 \vdash x(y, a.z) : [\rho_1, \rho_2]} \qquad w : [\sigma, \sigma] \Vdash w : [\sigma, \sigma] \qquad \Psi}{\Gamma_1 \vdash x(y, a.z)(w, b.b(b, c.c)) : \tau}$$

where $\Gamma_1 = z : [\rho_1, \rho_2]; w : [\sigma, \sigma]; x : [\sigma_1, \sigma_1]; y : [\sigma_2, \sigma_2]$.

While for the term t_2, we have:

$$\dfrac{x : [\sigma_1] \Vdash x : [\sigma_1] \qquad y : [\sigma_2] \Vdash y : [\sigma_2] \qquad \Phi}{\Gamma_2 \vdash x(y, a.z(w, b.b(b, c.c))) : \tau}$$

where

$$\Phi = \dfrac{z : [\rho_1, \rho_2] \Vdash z : [\rho_1, \rho_2] \qquad w : [\sigma, \sigma] \Vdash w : [\sigma, \sigma] \qquad \Psi}{\Gamma_2 \vdash z(w, b.b(b, c.c)) : \tau}$$

and $\Gamma_2 = z : [\rho_1, \rho_2]; w : [\sigma, \sigma]; x : [\sigma_1]; y : [\sigma_2]$.

Thus, the multiset types of x and y in Γ_1 and Γ_2 resp. are not the same. Despite the fact that the step $t_1 \to_\pi t_2$ does not erase any subterm, the typing environment is losing quantitative information.

Notice that by replacing non-idempotent types by idempotent ones, subject reduction (and expansion) would work for π-reduction: by assigning sets to variables instead of multisets, Γ_1 and Γ_2 would now represent the same object.

Despite the fact that quantitative subject reduction fails for some π-steps, the following weaker property is sufficient to recover (qualitative) soundness of our typing system $\cap J$ w.r.t. the reduction relation $\to_{\beta,\pi}$. Soundness will be used later in Sec. 6 to show equivalence between $\mathcal{SN}(\mathtt{d}\beta)$ and $\mathcal{SN}(\beta, \pi)$.

Lemma 14 (Typing Behavior of π-Reduction). *Let $\Gamma \Vdash^{n_1}_{\cap J} t_1 : \sigma$. If $t_1 = t(u, x.r)(u', y.r') \mapsto_\pi t_2 = t(u, x.r(u', y.r'))$, then there are n_2 and $\Sigma \sqsubseteq \Gamma$ such that $\Sigma \Vdash^{n_2}_{\cap J} t_2 : \sigma$ with $n_1 \geq n_2$.*

Lemma 15 (Soundness for ΛJ). *If t is $\cap J$-typable, then $t \in \mathcal{SN}(\beta, \pi)$.*

5 Faithfulness of the Translation

As discussed in the introduction, the natural translation [4] of generalized applications into ES is not faithful. In this section we define an alternative encoding and prove it faithful: a term in $\mathtt{T}_J$ is $\mathtt{d}\beta$-strongly normalizing *iff* its alternative encoding is strongly normalizing in the ES framework. In a later subsection, we use this connection with ES to establish the equivalence between strong normalization w.r.t. $\mathtt{d}\beta$ and $(\beta, \mathtt{p}_2)$.

5.1 Explicit Substitutions

We define the syntax and semantics of an ES calculus borrowed from [1] to which we relate λJ. It is a simple calculus where β is implemented in two independent steps: one creating a let-binding, and another one substituting the term bound. It has a notion of distance which allows to reduce redexes such as $([N/x](\lambda y.M))P \to_{\mathtt{dB}} [N/x][P/y]M$, where the ES $[N/x]$ lies between the abstraction and its argument. Terms and list contexts are given by:

$$\begin{array}{lr} (\mathtt{T}_{ES}) & M, N, P, Q ::= x \mid \lambda x.M \mid MN \mid [N/x]M \\ (\textbf{List contexts}) & \mathtt{L} ::= \Diamond \mid [N/x]\mathtt{L} \end{array}$$

The calculus λES is defined by $\mathtt{T}_{ES}[\mathtt{dB}, \mathtt{s}]$ (closed under all contexts) where:

$$\mathtt{L}\langle \lambda x.M \rangle N \mapsto_{\mathtt{dB}} \mathtt{L}\langle [N/x]M \rangle \qquad [N/x]M \mapsto_{\mathtt{s}} \{N/x\}M$$

Now, consider the (naive) translation from $\mathtt{T}_J$ to $\mathtt{T}_{ES}$ [4]:

$$x^\star := x \qquad (\lambda x.t)^\star := \lambda x.t^\star \qquad t(u, y.r)^\star := [t^\star u^\star/y]r^\star$$

According to this translation, the notion of distance in λES corresponds to our notion of distance for λJ. For instance, the application $t(u, x.\cdot)$ in the term $t(u, x.\lambda y.r)(u', z.r')$ can be seen as a substitution $[t^\star u^\star/x]\cdot$ inserted between the abstraction $\lambda y.r$ and the argument u'. But how can we now (informally) relate π to the notions of existing permutations for λES? Using the previous translation, we can see that $t_0 = t(u, x.r)(u', y.r') \mapsto_\pi t(u, x.r(u', y.r')) = t_1$ simulates as

$$t_0^\star = [([t^\star u^\star/x]r^\star)u'^\star/y]r'^\star \to [[t^\star u^\star/x](r^\star u'^\star)/y]r'^\star \to [t^\star u^\star/x][r^\star u'^\star/y]r'^\star = t_1^\star.$$

The first step is an instance of a rule in ES known as σ_1: $([u/x]t)v \mapsto [u/x](tv)$, and the second one of a rule we call σ_4: $[[u/x]t/y]v \mapsto [u/x][t/y]v$. Quantitative types for ES tell us that only rule σ_1, but not rule σ_4, is valid for a call-by-name calculus. This is why it is not surprising that π is rejected by our type system, as detailed in Sec. 4.3.

The alternative encoding we propose is as follows (noted $\_^*$ instead of $\_^\star$):

Definition 4 (Translation from $\mathtt{T}_J$ to $\mathtt{T}_{ES}$).

$$x^* := x \qquad (\lambda x.t)^* := \lambda x.t^* \qquad t(u, x.r)^* := [t^*/x^l][u^*/x^r]\{x^l x^r/x\}r^*$$

Notice the above π-reduction $t_0 \to t_1$ is still simulated: $t_0^* \to^2_{\sigma_4} t_1^*$.

Consider again the counterexample to faithfulness already discussed in the introduction, given by $t := \delta^\circ(\delta^\circ, y.r)$ with $y \notin \mathrm{fv}(r)$, where $\delta^\circ = \lambda x.x(x, w.w)$. The term t is a $\mathtt{d}\beta$-redex, whose contraction throws away the two copies of δ°. The naive translation of t gives $[\delta^{\circ *}\delta^{\circ *}/y]r^\star$, which clearly diverges in λES. The alternative encoding of t is $[\delta^{\circ *}/y^{\mathrm{l}}][\delta^{\circ *}/y^{\mathrm{r}}]\{y^{\mathrm{l}}y^{\mathrm{r}}/y\}r^*$, which is just $[\delta^{\circ *}/y^{\mathrm{l}}][\delta^{\circ *}/y^{\mathrm{r}}]r^*$, because $y \notin \mathrm{fv}(r^*)$. The only hope to have an interaction between the two copies of $\delta^{\circ *}$ in the previous term is to execute the ES, but such executions will just throw away those two copies, because $y^{\mathrm{l}}, y^{\mathrm{r}} \notin \mathrm{fv}(r^*)$. This gives an intuitive idea of the faithfulness of our encoding.

5.2 Proof of Faithfulness

We need to prove the equivalence between two notions of strong normalization: the one of a term in λJ and the one of its encoding in λES. While this proof can be a bit involved using traditional methods, quantitative types will make it very straightforward. Indeed, since quantitative types correspond exactly to strong normalization, we only have to show that a term t is typable exactly when its encoding is typable, for two appropriate quantitative type systems.

For λES, we will use the following system [9]:

Definition 5 (The Type System $\cap ES$).

$$\frac{}{x : [\sigma] \vdash x : \sigma}\ (\mathtt{var}) \qquad \frac{\Gamma; x : \mathcal{M} \vdash t : \sigma}{\Gamma \vdash \lambda x.M : \mathcal{M} \to \sigma}\ (\mathtt{abs})$$

$$\frac{(\Gamma_i \vdash M : \sigma_i)_{i\in I} \quad I \neq [\,]}{\wedge_{i\in I}\Gamma_i \vdash M : [\sigma_i]_{i\in I}}\ (\mathtt{many}) \qquad \frac{\Gamma \vdash M : \mathcal{M} \to \sigma \qquad \Delta \vdash N : \#(\mathcal{M})}{\Gamma \wedge \Delta \vdash MN : \sigma}\ (\mathtt{app})$$

$$\frac{\Gamma; x : \mathcal{M} \vdash M : \sigma \qquad \Delta \vdash N : \#(\mathcal{M})}{\Gamma \wedge \Delta \vdash [N/x]M : \sigma}\ (\mathtt{sub})$$

Theorem 4. *Let $M \in \mathtt{T}_{ES}$. Then M is typable in $\cap ES$ iff $M \in \mathcal{SN}(\mathtt{dB}, \mathtt{s})$.*

A simple induction on the type derivation shows that the encoding is sound.

Lemma 16. *Let $t \in \mathtt{T}_J$. Then $\Gamma \Vdash_{\cap J} t : \sigma \implies \Gamma \Vdash_{\cap ES} t^* : \sigma$.*

We show completeness by a detour through the encoding of $\mathtt{T}_{ES}$ to $\mathtt{T}_J$:

Definition 6 (Translation from $\mathtt{T}_{ES}$ to $\mathtt{T}_J$).

$$\begin{array}{rlrl} x^\circ &:= x & (MN)^\circ &:= M^\circ(N^\circ, x.x) \\ (\lambda x.M)^\circ &:= \lambda x.M^\circ & ([N/x]M)^\circ &:= \mathtt{I}(N^\circ, x.M^\circ) \end{array}$$

The two following lemmas, shown by induction on the type derivations, give in particular that t^* typable implies t typable.

Lemma 17. *Let $M \in \mathtt{T}_{ES}$. Then $\Gamma \Vdash_{\cap ES} M : \sigma \implies \Gamma \Vdash_{\cap J} M^\circ : \sigma$.*

Lemma 18. *Let $t \in \mathtt{T}_J$. Then $\Gamma \Vdash_{\cap J} t^{*\circ} : \sigma \implies \Gamma \Vdash_{\cap J} t : \sigma$.*

Putting all together, we get this equivalence:

Corollary 1. *Let $t \in \mathtt{T}_J$. Then $\Gamma \Vdash_{\cap J} t : \sigma \iff \Gamma \Vdash_{\cap ES} t^* : \sigma$.*

This corollary, together with the two characterization theorems 3 and 4, provides the main result of this section:

Theorem 5 (Faithfulness). *Let $t \in \mathtt{T}_J$. Then $t \in \mathcal{SN}(\mathtt{d}\beta) \iff t^* \in \mathcal{SN}(\mathtt{dB}, \mathtt{s})$.*

6 Equivalent Notions of Strong Normalization

In the previous section, we related strong $\mathtt{d}\beta$-normalization with strong normalization of ES. In this section we will compare the various concepts of strong normalization that are induced on $\mathtt{T}_J$ by β, $\mathtt{d}\beta$, $(\beta, \mathtt{p_2})$ and (β, π). This comparison will make use of several results obtained in the previous sections, and will obtain new results about the original calculus ΛJ.

6.1 β-Normalization Is Not Enough

We discussed in Sec. 2.2 about the unblocking property of π and $\mathtt{p_2}$. From the point of view of normalization, this means that $\mathtt{T}_J[\beta]$ has *premature* normal forms and that $\mathcal{SN}(\beta) \subsetneq \mathcal{SN}(\mathtt{d}\beta)$. To illustrate this purpose we give an example of a $\mathtt{T}_J$-term which normalizes when only using rule β, but diverges when adding permutation rules or distance. We write Ω the term $\delta^\circ(\delta^\circ, x.x)$, where $\delta^\circ = \lambda y.y(y, z.z)$, so that $\Omega \to_\beta \Omega$. Now, let us take $t := w(u, w'.\delta^\circ)(\delta^\circ, x.x)$. Although this term is normal in $\mathtt{T}_J[\beta]$, the second δ° is actually an argument for the first one, as we can see with a π permutation:

$$t \to_\pi w(u, w'.\delta^\circ(\delta^\circ, x.x)) = w(u, w'.\Omega) := t'$$

Thus $t \to_\pi t' \to_\beta t'$ which implies $t \notin \mathcal{SN}(\beta, \pi)$. We can also unblock the redex in t by a $\mathtt{p_2}$-permutation moving the inner λx up:

$$t \to_{\mathtt{p_2}} (\lambda y.w(u, w'.y(y, z.z)))(\delta^\circ, x.x) \to_\beta t'$$

Thus $t \to_{\mathtt{p_2}}\to_\beta t' \to_\beta t'$ and thus $t \notin \mathcal{SN}(\beta, \mathtt{p_2})$. We get the same thing in a unique $\mathtt{d}\beta$-step: $t \to_{\mathtt{d}\beta} t'$.

In all the three cases, β-strong normalization is not preserved by the permutation rules, as there is a term $t \in \mathcal{SN}(\beta)$ such that $t \notin \mathcal{SN}(\beta, \pi)$, $t \notin \mathcal{SN}(\beta, \mathtt{p_2})$ and $t \notin \mathcal{SN}(\mathtt{d}\beta)$.

6.2 Comparison with $\beta + \mathtt{p}_2$

We now formalize the fact that our calculus $\mathtt{T}_J[\mathtt{d}\beta]$ is a version with distance of $\mathtt{T}_J[\beta, \mathtt{p}_2]$, so that they are equivalent from a normalization point of view. For this, we will establish the equivalence between strong normalization w.r.t. $\mathtt{d}\beta$ and $(\beta, \mathtt{p}_2)$, through a long chain of equivalences. One of them is Thm. 5, that we have proved in the previous section; the other is a result about σ-rules in the λ-calculus – which is why we have to go through the λ-calculus again.

Definition 7 (Translation from $\mathtt{T}_{ES}$ to $\mathtt{T}_\lambda$).

$$x^\sharp := x \quad (\lambda x.M)^\sharp := \lambda x.M^\sharp \quad (MN)^\sharp := M^\sharp N^\sharp \quad [N/x]M^\sharp := (\lambda x.M^\sharp)N^\sharp$$

Lemma 19. *Let $M \in \mathtt{T}_{ES}$. Then $M \in \mathcal{SN}(\mathtt{dB}, \mathtt{s}) \implies M^\sharp \in \mathcal{SN}(\beta)$.*

Proof. For typability in the λ-calculus, we use the type system $\mathcal{S}'_\lambda$ with choice operators in [9], which we rename here $\cap S$. It can be seen as a restriction of our system $\cap ES$ to λ-terms. Suppose $M \in \mathcal{SN}(\mathtt{dB}, \mathtt{s})$. By Thm. 4 M is typable in $\cap ES$, and it is straightforward to show that $M^\sharp$ is typable in $\cap S$. Moreover, $M^\sharp$ typable implies that $M^\sharp \in \mathcal{SN}(\beta)$ [9], which is what we want.

For $t \in \mathtt{T}_J$, let $t^\square := t^{*\sharp}$. So, we are just composing the alternative encoding of generalized application into ES with the map into λ-calculus just introduced. The λ-term $t^\square$ may be given by recursion on t as follows:

$$x^\square = x \qquad (\lambda x.t)^\square = \lambda x.t^\square \qquad t(u, y.r)^\square = (\lambda y^{\mathrm{r}}.(\lambda y^{\mathrm{l}}.\{y^{\mathrm{l}}y^{\mathrm{r}}/y\}r^\square)t^\square)u^\square$$

Lemma 20. *$t^\square \in \mathcal{SN}(\beta, \sigma_2) \implies t \in \mathcal{SN}(\beta, \mathtt{p}_2)$.*

Proof. Because $(\cdot)^\square$ produces a strict simulation from $\mathtt{T}_J$ to $\mathtt{T}_\lambda$. More precisely: (i) if $t_1 \to_\beta t_2$ then $t_1^\square \to_\beta^+ t_2^\square$; (ii) if $t_1 \to_{\mathtt{p}_2} t_2$ then $t_1^\square \to_{\sigma_2}^2 t_2^\square$.

Theorem 6. *Let $t \in \mathtt{T}_J$. Then $t \in \mathcal{SN}(\beta, \mathtt{p}_2)$ iff $t \in \mathcal{SN}(\mathtt{d}\beta)$.*

Proof. We prove that the following conditions are equivalent: 1) $t \in \mathcal{SN}(\beta, \mathtt{p}_2)$. 2) $t \in \mathcal{SN}(\mathtt{d}\beta)$. 3) $t^* \in \mathcal{SN}(\mathtt{dB}, \mathtt{s})$. 4) $t^\square \in \mathcal{SN}(\beta)$. 5) $t^\square \in \mathcal{SN}(\beta, \sigma_2)$. Now, 1) $\implies$ 2) is because $\to_{\mathtt{d}\beta} \subset \to_{\beta,\mathtt{p}_2}^+$. 2) $\implies$ 3) is by Thm. 5. 3) $\implies$ 4) is by Lem. 19. 4) $\implies$ 5) is showed in [13]. 5) $\implies$ 1) is by Lem. 20.

6.3 Comparison with $\beta + \pi$

We now prove the equivalence between strong normalization for $\mathtt{d}\beta$ and for (β, π). One of the implications already follows from the properties of the typing system.

Lemma 21. *Let $t \in \mathtt{T}_J$. If $t \in \mathcal{SN}(\mathtt{d}\beta)$ then $t \in \mathcal{SN}(\beta, \pi)$.*

Proof. Follows from the completeness of the typing system (Lem. 13) and soundness of $\cap J$ for (β, π) (Lem. 15).

$$\frac{}{x \in \mathcal{ISN}(\beta,\pi)}\,(var) \qquad \frac{u, r \in \mathcal{ISN}(\beta,\pi)}{x(u, z.r) \in \mathcal{ISN}(\beta,\pi)}\,(hvar) \qquad \frac{t \in \mathcal{ISN}(\beta,\pi)}{(\lambda x.t) \in \mathcal{ISN}(\beta,\pi)}\,(lambda)$$

$$\frac{x(u, y.rS)\boldsymbol{S} \in \mathcal{ISN}(\beta,\pi)}{x(u, y.r)S\boldsymbol{S} \in \mathcal{ISN}(\beta,\pi)}\,(pi) \qquad \frac{\{\{u/x\}t/y\}r\boldsymbol{S} \in \mathcal{ISN}(\beta,\pi) \quad t, u \in \mathcal{ISN}(\beta,\pi)}{(\lambda x.t)(u, y.r)\boldsymbol{S} \in \mathcal{ISN}(\beta,\pi)}\,(beta)$$

Fig. 1. Inductive characterization of the strong (β,π)-normalizing ΛJ-terms

The proof of the other implication requires more work, organized in 4 parts: 1) A remark about ES. 2) A remark about translations of ES into the ΛJ-calculus. 3) Two new properties of strong normalization for (β,π) in ΛJ. 4) Preservation of strong (β,π)-normalization by a certain map from the set $\mathtt{T}_J$ into itself.

The remark about explicit substitutions is this:

Lemma 22. *For all $M \in \mathtt{T}_{ES}$, $M \in \mathcal{SN}(\mathtt{dB},\mathtt{s})$ iff $M \in \mathcal{SN}(\mathtt{B},\mathtt{s})$.*

The translation $\_^\circ$ in Def. 6 induces a simulation of each $\mathtt{s}$-reduction step on $\mathtt{T}_{ES}$ into a β-reduction step on $\mathtt{T}_J$, but cannot simulate the creation of an ES effected by rule $\mathtt{B}$. A solution is to refine the translation $\_^\circ$ for applications, yielding the following alternative translation:

$$\begin{array}{rlrl} x^\bullet & := x & (\lambda x.M)^\bullet & := \lambda x.M^\bullet \\ (MN)^\bullet & := I(N^\bullet, y.M^\bullet(y, z.z)) & [N/x]M^\bullet & := I(N^\bullet, x.M^\bullet) \end{array}$$

Since the clause for ES is not changed, simulation of each $\mathtt{s}$-reduction step by a β-reduction step holds as before. The improvement lies in the simulation of each $\mathtt{B}$-reduction step:

$$((\lambda x.M)N)^\bullet = I(N^\bullet, y.(\lambda x.M^\bullet)(y, z.z)) \to_\beta I(N^\bullet, y.\{y/x\}M^\bullet) =_\alpha ([N/x]M)^\bullet$$

This strict simulation gives immediately:

Lemma 23. *For all $M \in \mathtt{T}_{ES}$, if $M^\bullet \in \mathcal{SN}(\beta)$ then $M \in \mathcal{SN}(\mathtt{B},\mathtt{s})$.*

We now prove two properties of strong normalization for (β,π) in ΛJ. Following [10], $\mathcal{SN}(\beta,\pi)$ admits an inductive characterization $\mathcal{ISN}(\beta,\pi)$, given in Fig. 1, which uses the following inductive generation for $\mathtt{T}_J$-terms:

$$t, u, r \ ::= \ x\boldsymbol{S} \mid \lambda x.t \mid (\lambda x.t)S\boldsymbol{S} \qquad S \ ::= \ (u, y.r)$$

Hence S stands for a *generalized* argument, while $\boldsymbol{S}$ denotes a possibly empty list of S's. Notice that at most one rule in Fig. 1 applies to a given term, so the rules are deterministic (and thus invertible).

A preliminary fact is the following:

Lemma 24. *$\mathcal{SN}(\beta,\pi)$ is closed under prefixing of arbitrary π-reduction steps:*

$$\frac{t \to_\pi t' \text{ and } t' \in \mathcal{SN}(\beta,\pi)}{t \in \mathcal{SN}(\beta,\pi)}$$

Given that $\mathcal{SN}(\beta,\pi)=\mathcal{ISN}(\beta,\pi)$, the "rule" in Lem. 24, when written with $\mathcal{ISN}(\beta,\pi)$, is admissible for the predicate $\mathcal{ISN}(\beta,\pi)$. Now, consider:

$$\frac{u, r \in \mathcal{ISN}(\beta,\pi)}{\{y(u, z.z)/x\}r \in \mathcal{ISN}(\beta,\pi)}\,(I)$$

$$\frac{\{\{\{u/y\}t/z\}r/x\}r \in \mathcal{ISN}(\beta,\pi) \qquad t, u \in \mathcal{ISN}(\beta,\pi) \qquad x \notin \mathrm{fv}(t,u,r)}{\{(\lambda y.t)(u, z.r)/x\}r \in \mathcal{ISN}(\beta,\pi)}\,(II)$$

Notice rule II generalizes rule (*beta*) in Fig. 1: just take $r = x\boldsymbol{S}$, with $x \notin \boldsymbol{S}$.

The two new properties of strong normalization for (β,π) in ΛJ are contained in the following Lemma.

Lemma 25. *Rules I and II are admissible rules for the predicate $\mathcal{ISN}(\beta,\pi)$.*

We now move to the fourth part of the ongoing reasoning. Consider the map from $\mathtt{T}_J$ to itself obtained by composing $(\cdot)^* : \mathtt{T}_J \to \mathtt{T}_{ES}$ with $(\cdot)^\bullet : \mathtt{T}_{ES} \to \mathtt{T}_J$. Let us write $t^\dagger := t^{*\bullet}$. A recursive definition is also possible, as follows:

$$x^\dagger = x \qquad \lambda x.t^\dagger = \lambda x.t^\dagger \qquad t(u, y.v)^\dagger = I(t^\dagger, y_1.I(u^\dagger, y_2.\{y_1(y_2, z.z)/y\}v^\dagger))$$

Lemma 26. *If $t \in \mathcal{SN}(\beta,\pi)$ then $t^\dagger \in \mathcal{SN}(\beta,\pi)$.*

Proof. Heavy use is made of Lem. 24 and Lem. 25.

All is in place to obtain the desired result:

Theorem 7. *Let $t \in \mathtt{T}_J$. $t \in \mathcal{SN}(\mathtt{d}\beta)$ iff $t \in \mathcal{SN}(\beta,\pi)$.*

Proof. The implication from left to right is Lem. 21. For the converse, suppose $t \in \mathcal{SN}(\beta,\pi)$. By Lem. 26, $t^\dagger \in \mathcal{SN}(\beta,\pi)$. Trivially, $t^\dagger \in \mathcal{SN}(\beta)$. Since $t^\dagger = t^{*\bullet}$, Lem. 23 gives $t^* \in \mathcal{SN}(\mathtt{B},\mathtt{s})$. By Lem. 22, $t^* \in \mathcal{SN}(\mathtt{dB},\mathtt{s})$. By an equivalence in the proof of Thm. 6, $t \in \mathcal{SN}(\mathtt{d}\beta)$.

6.4 Consequences for ΛJ

The comparison with λJ gives new results about the original ΛJ (a quantitative typing system characterizing strong normalization, and a faithful translation into ES) as immediate consequences of Thms. 3, 5, and 7.

Corollary 2. *Let $t \in \mathtt{T}_J$. (1) $t \in \mathcal{SN}(\beta,\pi)$ iff t is $\cap J$-typable. (2) $t \in \mathcal{SN}(\beta,\pi)$ iff $t^* \in \mathcal{SN}(\mathtt{dB},\mathtt{s})$.*

Beyond strong normalization, ΛJ gains a new normalizing strategy, which reuses the notion of weak-head normal form introduced in Sec. 3.2. We take the definitions of neutral terms, answer and weak-head context $\mathtt{W}$ given there for λJ, in order to define a new weak-head strategy and a new predicate ISN for ΛJ. The strategy is defined as the closure under $\mathtt{W}$ of rule β and of the particular case of rule π where the redex has the form $\mathtt{n}(u, x.\mathtt{a})S$[5].

[5] Notice how a redex has the two possible forms $(\lambda x.t)S$ or $\mathtt{n}(u, x.\mathtt{a})S$, that can be written as $\mathtt{a}S$, that is, the form $\mathtt{D_n}\langle\lambda x.t\rangle S$ of a weak-head redex in λJ

Definition 8. *Predicate ISN is defined by the rules* (snvar), (snapp), (snabs) *in Def. 3, together with the following two rules (which replace rule* (snbeta)*):*

$$\frac{\mathtt{W}\langle \mathtt{n}(u, y.\mathtt{a}S)\rangle \in ISN}{\mathtt{W}\langle \mathtt{n}(u, y.\mathtt{a})S\rangle \in ISN}\ (\mathtt{snredex1}) \qquad \frac{\mathtt{W}\langle \{\{u/x\}t/y\}r\rangle, t, u \in ISN}{\mathtt{W}\langle (\lambda x.t)(u, y.r)\rangle \in ISN}\ (\mathtt{snredex2})$$

The corresponding normalization strategy is organized as usual: an initial phase obtains a weak-head normal form, whose components are then reduced by internal reduction. Is this new strategy any good? The last theorem of the paper answers positively:

Theorem 8. *Let* $t \in \mathtt{T}_J$. $t \in ISN$ *iff* $t \in \mathcal{ISN}(\beta, \pi)$.

7 Conclusion

Contributions. This paper presents and studies several properties of the call-by-name λJ-calculus, a formalism implementing an appropriate notion of distant reduction to unblock the β-redexes arising in generalized application notation.

Strong normalization of simple typed terms was shown by translating the λJ into the λ-calculus. A full characterization of strong normalization was developed by means of a quantitative type system, where the length of $\mathtt{d}\beta$-reduction to normal form is bound by the size of the type derivation of the starting term. An inductive definition of $\mathtt{d}\beta$-strong normalization was defined and proved correct in order to achieve this characterization. It was also shown how the traditional permutative rule π is rejected by the quantitative system, thus emphasizing the choice of $\mathtt{d}\beta$-reduction for a quantitative generalized application framework.

We have also defined a faithful translation from the λJ-calculus into ES. The translation preserves strong normalization, in contrast to the traditional translation to ES *e.g.* in [4]. Last but not least, we related strong normalization of λJ with that of other calculi, including in particular the original ΛJ. New results for the latter were found by means of the techniques developed for λJ. In particular, a quantitative characterization of strong normalization was developed for ΛJ, where the bound of reduction given by the size of type derivations only holds for β-steps (and not for π-steps).

Future work. Regarding call-by-name for generalized applications, this paper opens new questions. We studied a new calculus λJ, proposed as an alternative to the original ΛJ, but we also mentioned some possible variants in Sec. 2.2, notably a calculus based on rule (1), and $\beta + \mathtt{p}_2$ (used as a technical tool in Sec. 6). The first option seems to have the flavor of ΛJ whereas the $\beta + \mathtt{p}_2$ option seems to have the flavor of λJ. It remains to be seen what are the advantages and drawbacks of the latter one with respect to λJ.

Regarding call-by-value, we plan to develop the quantitative semantics in the presence of generalized applications, starting from the calculus proposed in [5]. Further unification between call-by-name and call-by-value with the help of generalized applications could be considered in the setting of the polarized lambda-calculus [4].

References

1. Accattoli, B.: An abstract factorization theorem for explicit substitutions. In: Tiwari, A. (ed.) 23rd International Conference on Rewriting Techniques and Applications (RTA'12) , RTA 2012, May 28 - June 2, 2012, Nagoya, Japan. LIPIcs, vol. 15, pp. 6–21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2012)
2. Accattoli, B., Kesner, D.: The structural λ-calculus. In: Dawar, A., Veith, H. (eds.) Computer Science Logic, 24th International Workshop, CSL 2010, 19th Annual Conference of the EACSL, Brno, Czech Republic, August 23-27, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6247, pp. 381–395. Springer (2010)
3. Espírito Santo, J., Frade, M.J., Pinto, L.: Structural proof theory as rewriting. In: Pfenning, F. (ed.) Term Rewriting and Applications. pp. 197–211. Lecture Notes in Computer Science, Springer (2006)
4. Espírito Santo, J.: Delayed substitutions. In: Lecture Notes in Computer Science, pp. 169–183. Springer Berlin Heidelberg (2007)
5. Espírito Santo, J.: The call-by-value lambda-calculus with generalized applications. In: CSL. LIPIcs, vol. 152, pp. 35:1–35:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2020)
6. Espírito Santo, J., Kesner, D., Peyrot, L.: A faithful and quantitative notion of distant reduction for generalized applications. CoRR **abs/2201.04156** (2022)
7. Espírito Santo, J., Pinto, L.: Permutative conversions in intuitionistic multiary sequent calculi with cuts. In: Hofmann, M. (ed.) Typed Lambda Calculi and Applications. pp. 286–300. Lecture Notes in Computer Science, Springer (2003)
8. Joachimski, F., Matthes, R.: Standardization and confluence for a lambda calculus with generalized applications. In: Rewriting Techniques and Applications, pp. 141–155. Springer Berlin Heidelberg (2000)
9. Kesner, D., Vial, P.: Non-idempotent types for classical calculi in natural deduction style. Log. Methods Comput. Sci. **16**(1) (2020)
10. Matthes, R.: Characterizing strongly normalizing terms of a calculus with generalized applications via intersection types. In: Rolim, J.D.P., Broder, A.Z., Corradini, A., Gorrieri, R., Heckel, R., Hromkovic, J., Vaccaro, U., Wells, J.B. (eds.) ICALP Workshops 2000, Proceedings of the Satelite Workshops of the 27th International Colloquium on Automata, Languages and Programming, Geneva, Switzerland, July 9-15, 2000. pp. 339–354. Carleton Scientific, Waterloo, Ontario, Canada (2000)
11. von Plato, J.: Natural deduction with general elimination rules. Arch. Math. Log. **40**(7), 541–567 (2001)
12. van Raamsdonk, F.: Confluence and normalisation for higher-order rewriting. Ph.D. thesis, Vrije Universiteit Amsterdam (May 1996)
13. Regnier, L.: Une équivalence sur les lambda-termes. Theor. Comput. Sci. **126**(2), 281–292 (1994)