A FAITHFUL AND QUANTITATIVE NOTION OF DISTANT REDUCTION FOR GENERALIZED APPLICATIONS

JOSÉ ESPÍRITO SANTO a, DELIA KESNER b,c, AND LOÏC PEYROT b

a Universidade do Minho, Braga, Portugal
e-mail address: jes@math.uminho.pt

b Université Paris Cité, CNRS, IRIF, Paris, France
e-mail address: kesner@irif.fr, lpeyrot@irif.fr

c Institut Universitaire de France

Abstract. We introduce a call-by-name lambda-calculus $\lambda J_n$ with generalized applications which is equipped with distant reduction. This allows to unblock $\beta$-redexes without resorting to the standard permutative conversions of generalized applications used in the original $\Lambda J$-calculus with generalized applications of Joachimski and Matthes. We show strong normalization of simply-typed terms, and we then fully characterize strong normalization by means of a quantitative (i.e. non-idempotent intersection) typing system. This characterization uses a non-trivial inductive definition of strong normalization –related to others in the literature–, which is based on a weak-head normalizing strategy. We also show that our calculus $\lambda J_n$ relates to explicit substitution calculi by means of a faithful translation, in the sense that it preserves strong normalization. Moreover, our calculus $\lambda J_n$ and the original $\Lambda J$-calculus determine equivalent notions of strong normalization. As a consequence, $\Lambda J$ inherits a faithful translation into explicit substitutions, and its strong normalization can also be characterized by the quantitative typing system designed for $\lambda J_n$, despite the fact that quantitative subject reduction fails for permutative conversions.

1. Introduction

In the original calculus with generalized applications $\Lambda J$, due to Joachimski and Matthes [JM03, JM00], the standard syntax of the $\lambda$-calculus is modified by generalizing the application constructor $tu$ into a new shape $t(u, y.r)$, capturing a notion of sharing for applications: a term $t(u, y.r)$ can intuitively be understood as a let-binding of the form let $y = tu$ in $r$.

This new constructor can better be understood in a typed framework. Indeed, the simply-typed $\Lambda J$-calculus is an interpretation of the implicative fragment of von Plato’s system of natural deduction with generalized elimination rules [vP01] under the Curry-Howard correspondence. Besides the logical reading, the syntax with generalized applications constitutes also a minimal framework for studying the call-by-name (CbN) and call-by-value (CbV) functional paradigms, as well as various kinds of permutative conversions beyond the $\lambda$-calculus.
The operational semantics of $\Lambda J$ is given by a call-by-name $\beta$-rule generalizing the one of the $\lambda$-calculus, as well as a permutative $\pi$-rule on terms. The two rules are as follows:

\[
(\lambda x.t)(u, z.r) \rightarrow_\beta \{u/x\}t/zr \\
t(u, y.r)(u', z.r') \rightarrow_\pi t(u, y.r(u', z.r'))
\]

In a typed setting, the reduction of terms in $\Lambda J$ corresponds to normalization in natural deduction with generalized elimination rules. The $\pi$-rule corresponds to a commutative conversion caused by a repetition of a same formula as premise and conclusion of an elimination of the associated logical system: in the redex above, the type of $t(u, y.r)$ is the same as the type of $r$ (this is analogous to what happens with the elimination rule for disjunction). Such normalization process brings proofs to the so-called fully normal form. Fully-normal proofs enjoy the subformula property and are in one-to-one correspondence with the cut-free derivations of the sequent calculus [vP01].

Both reduction rules make perfect sense in the untyped setting as well. The $\beta$-rule executes the call of the function $\lambda x.t$ with argument $u$: this eliminates the shared application, and the result $\{u/x\}u$ has to be unshared within the continuation $r$. The $\pi$-rule implements a permutation allowing to unblock $\beta$-redexes. For example, in the reduction sequence $t(u, y.\lambda x.r)(u', z.r') \rightarrow_\pi t(u, y.(\lambda x.r)(u', z.r')) \rightarrow_\beta t(u, y.\{u'/x\}r/zr')$, the first step reveals a $\beta$-redex, previously hidden. Indeed, some $\beta$-redexes are obstructed by the syntax of terms, and rearrangement of terms through $\pi$-reduction steps is often necessary to achieve fully normal forms.

Strong normalization w.r.t. the two rules $\beta$ and $\pi$ has been characterized by typability in an (idempotent) intersection typing system by Matthes [Mat00]: 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 [Gar94], 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 $t$, as well as the size of its normal form, is bound by the size of its type derivation. Non-idempotent intersection (a.k.a. quantitative) types are an inductive representation of the relational models of linear logic [dC07]. 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, as we will observe, in the $\Lambda J$-calculus rule $\pi$ is not quantitatively sound (i.e. $\pi$ does not enjoy quantitative subject reduction), although $\pi$ becomes valid in a qualitative (idempotent) framework. Hence, one of the first questions that arises is: How can we unblock redexes to reach normal forms in a quantitative model of computation based on generalized applications?

Our solution relies on a different permutation rule $t(u, y.\lambda z.r) \rightarrow_{p2} \lambda z.t(u, y.r)$. More precisely, instead of considering rule $p2$ independently from $\beta$, we adopt the paradigm of distant reduction [AK10], which extends the key concept of $\beta$-redex, so that it is possible to find a $\lambda$-abstraction hidden under a certain context (in our case under a sequence of nested generalized applications). To do so, we directly integrate the $p2$-permutations that are necessary to unblock reduction together with $\beta$, creating a distant $\beta$ rule, called $d\beta$.

1A recent call-by-value variant, proposed in [ES20], is out of the scope of this paper.

2Rule $p2$ is used in [ESP03, ESP11] along with two other permutation rules $p1$ and $p3$ to reduce terms with generalized applications to a form corresponding to ordinary $\lambda$-terms.
This choice does not affect (strong) normalization, which is our focus, and highlights the computational behavior of the calculus: it is at the $\beta$-step that resources are consumed, not during the permutations.

The syntax of the $\Lambda J$-calculus will thus be equipped with an operational call-by-name semantics given by the distant rule $d\beta$, but without $\pi$. The resulting calculus is called $\lambda J_n$. 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 $d\beta$-step. For the completeness result (strong normalization implies typability) we need an inductive characterization of the terms that are strongly normalizing for $d\beta$: this is a non-trivial technical contribution of the paper.

Our new calculus $\lambda J_n$ is then compatible with a quantitative typing system. However, this type system designed for $\lambda J_n$ only partially captures strong normalization for $\Lambda J$ on a quantitative level, because the bound for reduction lengths given by the size of type derivations only holds for $\beta$, and not for $\pi$. Nevertheless, using this partial bound, we can prove that the type system designed for $\lambda J_n$ is also sound for strong normalization in the original calculus $\Lambda J$, in the sense that any typable term is strongly normalizing. It immediately follows that a term $t$ is strongly normalizable in $\lambda J_n$ only if $t$ is strongly normalizable in $\Lambda J$.

Actually, we go further and prove that this implication is an equivalence. The central role in the proof is again played by intersection type systems, together with a new encoding of generalized applications into explicit substitutions (ES). More precisely, we consider a calculus with explicit substitutions, where a new constructor $[u/x]t$, akin to a let-binding let $x = u$ in $t$, is added to the grammar of the $\lambda$-calculus. The reading given above of $t(u, y.r)$ as a let-binding expressing the sharing of the application $tu$ is similar to the intuitive and known [ES07] translation of $t(u, y.r)$ into the explicit substitution $[tu/y]r$. This translation, however, does not suit our goals, because it does not preserve strong normalization: a non-terminating computation generated by the interaction of $t$ with $u$ in $t(u, y.r)$ will always have to be substituted for $y$ in $r$, and thus may vanish if $y$ does not occur free in $r$ (a detailed example will be given later). We instead propose a new, type-preserving encoding of terms with generalized applications into ES, and show the dynamic behavior of our calculus $\lambda J_n$ to be faithful to explicit substitutions: a term is strongly normalizing in $\lambda J_n$ if and only if its new encoding into ES is also strongly normalizing. The proof of faithfulness essentially relies on an analysis of typability in the type system designed for $\lambda J_n$.

**Plan of the paper.** Section 2 presents and motivates our calculus $\lambda J_n$ with distant $\beta$. Section 3 provides an inductive characterization of strongly normalizing terms in $\lambda J_n$. Section 4 presents the non-idempotent intersection type system for $\lambda J_n$, proves the characterization of strong normalization in $\lambda J_n$ as typability in that system, and discusses why $\pi$ is not quantitative. Section 5 defines the new translation into ES and proves it to be faithful, in the sense of preserving and reflecting strong normalization. Section 6 contains comparisons with other calculi, obtained by equipping the terms of $\lambda J_n$ with $\beta$, $\beta + p2$, and $\beta + \pi$. The main focus there is to prove the respective notions of strong normalization equivalent, but we also collect the results $\Lambda J$ inherits from our study of $\lambda J_n$. Section 7 summarizes our contributions and discusses future work.
2. A calculus with generalized applications

In this section we define our calculus with generalized applications, denoted λJₙ. Starting from the issue of stuck redexes, we discuss different possibilities for the operational semantics. Next we prove some introductory properties of the calculus we propose.

2.1. Syntax. We start with some general notations. A reduction rule, denoted \( \mathcal{R} \) or \( \to \mathcal{R} \), is a binary relation in the syntax of some calculus, and generates a reduction relation \( \to \mathcal{R} \), usually by closure of \( \mathcal{R} \) under all contexts. 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 normalising (written \( t \in SN(\mathcal{R}) \)) iff there is no infinite \( \mathcal{R} \)-reduction sequence starting at \( t \). \( \mathcal{R} \) is strongly normalising iff every term is \( \mathcal{R} \)-strongly normalising. When \( \to \mathcal{R} \) 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 \( t \in SN(\mathcal{R}) \).

The set of terms generated by the following grammar is denoted by \( T_J \).

\[
(\text{Terms}) \quad t, u, r, s ::= x \mid \lambda x.t \mid t(u, x.r)
\]

We use \( I \) to denote the identity function \( \lambda z.z \) and \( \delta \) to denote the term \( \lambda x.x(x, z.z) \). The term \( t(u, x.r) \) is called a generalized application, and the part \( x.r \) is sometimes referred as the continuation of the generalized application. Free variables of terms are defined as usual, notably \( \text{fv}(t(u, x.r)) := \text{fv}(t) \cup \text{fv}(u) \cup \text{fv}(r) \setminus \{x\} \). We work modulo \( \alpha \)-conversion, denoted \( =_\alpha \), so that bound variables can be systematically renamed. The substitution operation is capture-avoiding and defined as usual, in particular \( \{u/x\}(t(s, y.r)) := \{(u/x)t\}(\{u/x\}s, y.\{u/x\}r) \), where \( y \notin \text{fv}(u) \).

Contexts (terms with one occurrence of the hole \( \diamond \) ) and distant contexts are given by the following grammars:

\[
(\text{Contexts}) \quad C ::= \diamond \mid \lambda x.C \mid C(u, x.r) \mid t(C, x.r) \mid t(u, x.C)
\]

\[
(\text{Distant contexts}) \quad D ::= \diamond \mid t(u, x.D)
\]

The term \( C(t) \) denotes \( C \) where \( \diamond \) is replaced by \( t \), so that capture of variables may eventually occur. We say that \( t \) has an abstraction shape iff \( t = D(\lambda x.u) \).

Given a reduction rule \( \mathcal{R} \subseteq T_J \times T_J, \to \mathcal{R} \) denotes the reduction relation generated by the closure of \( \to \mathcal{R} \) under all contexts. The syntax of \( T_J \) can be equipped with different rewriting rules. We use the generic notation \( T_J[\mathcal{R}] \) to denote the calculus given by the syntax \( T_J \) equipped with the reduction relation \( \to \mathcal{R} \). In particular, the \( \Lambda J \)-calculus \( [JM00] \) is given by \( T_J[\beta, \pi] \), where we recall \( \beta \) and \( \pi \), defined in section 1:

\[
(\lambda x.t)(u, x.z.r) \to^\beta \{u/x\}t/z)r
\]

\[
t(u, y.r)(u', z.r') \to^\pi t(u, y.\{u'/x\}s/z)r'
\]

2.2. Towards a Call-by-Name Operational Semantics. When considering \( t_0 := t(u, y.\lambda x.s)(u', z.r') \) in the calculus \( T_J[\beta] \), we can see that the term \( t_0 \) is stuck, since the subterm \( \lambda x.s \) is not close to its argument \( u' \) in order to fire a substitution. This is when the permutative rule \( \pi \) plays the role of an unblocker for \( \beta \)-redexes. Indeed,

\[
t_0 \to^\pi t(u, y.\{u'/x\}s/z)r'
\]
More generally, given $t := D(\lambda x.s)(u, y.r)$ with $D \neq \circ$, a sequence of $\pi$-steps reduces the term $t$ above to $D(\lambda x.s)(u, y.r)$. A further $\beta$-step produces $D(\{u/x\} s/y) r$. So, the original $\Lambda J$-calculus, which is exactly $T[J[\beta, \pi]]$, has an associated derived notion of distant $\beta$ rule, based on $\pi$. This rule $d\beta\pi$ is specified as follows.

$$D(\lambda x.s)(u, y.r) \mapsto\!\rightarrow_{d\beta\pi} D(\{u/x\} s/y) r$$

(2.1)

Still, the operational semantics that we propose in this paper will not reduce as in (2.1), because such rule, as well as $\pi$ itself, does not admit a quantitative semantics (see subsection 4.3). We then choose to unblock $\beta$-redexes with rule $p2$ instead:

$$t(u, y.\lambda x.s) \mapsto\!\rightarrow_{p2} \lambda x.t(u, y.s)$$

More precisely, we adopt rule $d\beta$, meaning distant $\beta$, which integrates $p2$ inside $\beta$:

$$D(\lambda x.t)(u, y.r) \mapsto\!\rightarrow_{d\beta} \{D(u)/x\} t/y) r$$

Note that since the free variables in $u$ cannot be captured by $D$, the right-hand term is equal to $\{D(\{u/x\} t)/y\} r$.

Definition 2.1. The distant calculus with generalized applications is given by $\lambda J_n := T[J[d\beta]]$.

Comparing the two rules $d\beta\pi$ and $d\beta$ gives a first intuition on why the first one is not quantitatively correct. In the rule $d\beta\pi$, the distant context is outside of the two substitutions: a unique copy is kept, which is independent from the number of occurrences of $y$ in $r$. This is a CbV behavior, neither erasing nor duplicating computations. On the contrary, the distant context may be erased or duplicated in rule $d\beta$ according to the number of occurrence of the variable $y$ in the term $r$.

In summary, $T[J[d\beta\pi]]$ does not provide a sound semantics for a resource-aware model, such as the one given by a quantitative type system. More precisely, quantitative subject reduction does not hold for (a rule relying on) $\pi$, as is shown in subsection 4.3. We adopt $T[J[d\beta]]$ instead.

2.3. Some Properties of $\lambda J_n$. In this section we discuss some properties of the new calculus, namely, characterization of $d\beta$-normal forms, the subformula property for simply typed terms, $d\beta$-strong normalization and confluence.

Characterization of Normal Forms. It is usually common to characterize the set of normal forms by means of a context free grammar.

Lemma 2.2. The grammar $NF_{d\beta}$ characterizes $d\beta$-normal forms.

$$NF_{d\beta} := x \mid \lambda x.NF_{d\beta} \mid NE_{d\beta}(NF_{d\beta}, x, NF_{d\beta})$$

$$NE_{d\beta} := x \mid NE_{d\beta}(NF_{d\beta}, x, NE_{d\beta})$$

Proof. We start with soundness: $t \in NF_{d\beta} \implies t$ is in $d\beta$-nf. We show the following two stronger properties:

(1) For all $t \in NE_{d\beta}$, $t$ does not have an abstraction shape and $t$ is in $d\beta$-nf.

(2) For all $t \in NF_{d\beta}$, $t$ is in $d\beta$-nf.

The proof is by simultaneous induction on $t \in NE_{d\beta}$ and $t \in NF_{d\beta}$.

First, the cases relative to the first item.

Case $t = x$: A variable $x$ does not have an abstraction shape and is in $d\beta$-nf.
Case \( t = s(u, x.r) \), with \( s, r \in \text{NE}_{d\beta} \) and \( u \in \text{NF}_{d\beta} \): The term \( t \) does not have an abstraction shape (because \( r \) does not have an abstraction shape, due to \( i.h. \) (1)). The term \( t \) is in \( d\beta\text{-nf} \) because \( s, u, r \) are in \( d\beta\text{-nf} \) (due to \( i.h. \) (1) and (2)) and because \( t \) itself is not a \( d\beta\text{-redex} \) (since \( s \) does not have an abstraction shape, by \( i.h. \) (1)).

Next, the cases relative to the second item.

Case \( t = x \): A variable \( x \) is in \( d\beta\text{-nf} \).

Case \( t = \lambda x.s \), with \( s \in \text{NF}_{d\beta} \): By \( i.h. \) (2), \( s \) is in \( d\beta\text{-nf} \). Hence so is \( \lambda x.s \).

Case \( t = s(u, x.r) \), with \( s \in \text{NE}_{d\beta} \) and \( u, r \in \text{NF}_{d\beta} \): The term \( t \) is in \( d\beta\text{-nf} \) because \( s, u, r \) are in \( d\beta\text{-nf} \) (due to \( i.h. \) (1), (2)) and because \( t \) itself is not a \( d\beta\text{-redex} \) (since \( s \) does not have an abstraction shape, by \( i.h. \) (1)).

Now, completeness: \( t \) is in \( d\beta\text{-nf} \) \( \implies \) \( t \in \text{NF}_{d\beta} \). We show a stronger property: For all \( t \),

(1) If \( t \) does not have an abstraction shape and \( t \) is in \( d\beta\text{-nf} \), then \( t \in \text{NE}_{d\beta} \); and

(2) If \( t \) is in \( d\beta\text{-nf} \), then \( t \in \text{NF}_{d\beta} \).

The proof is by induction on \( t \).

Case \( t = x \): We have \( x \in \text{NE}_{d\beta} \) and \( x \in \text{NF}_{d\beta} \).

Case \( t = \lambda x.s \): Part (1) is trivial. Suppose \( t \) is in \( d\beta\text{-nf} \). Then so is \( s \). By the \( i.h. \) (1), \( s \in \text{NF}_{d\beta} \). Hence \( \lambda x.s \in \text{NF}_{d\beta} \).

Case \( t = s(u, x.r) \): Suppose \( t \) is in \( d\beta\text{-nf} \). Then \( s, u, r \) are in \( d\beta\text{-nf} \), hence \( u \in \text{NF}_{d\beta} \) and \( r \in \text{NF}_{d\beta} \), by \( i.h. \) (1). The subterm \( s \) does not have an abstraction shape, otherwise \( t \) would be a \( d\beta\text{-redex} \), thus \( s \in \text{NE}_{d\beta} \), by the \( i.h. \) (1). Therefore, \( t \in \text{NF}_{d\beta} \) and (1) is proved. Moreover, suppose \( t \) does not have an abstraction shape. Then the same holds for \( r \). By \( i.h. \) (1) \( r \in \text{NE}_{d\beta} \). Hence \( t \in \text{NE}_{d\beta} \) and (1) is proved. \( \Box \)

We already saw that, once \( \beta \) is generalized to \( d\beta \), \( \pi \) is not needed anymore to unblock \( \beta\text{-redexes} \); the next lemma says that \( \pi \) preserves \( d\beta\text{-nfs} \), so it does not bring anything new to \( d\beta\text{-nfs} \) either.

**Lemma 2.3.** If \( t \) is a \( d\beta\text{-nf} \), and \( t \to \pi t' \), then \( t' \) is a \( d\beta\text{-nf} \).

**Proof.** Given Lemma 2.2, the proof proceeds by simultaneous induction on \( \text{NF}_{d\beta} \) and \( \text{NE}_{d\beta} \) (for \( \text{NE}_{d\beta} \) one also proves that \( \text{NE}_{d\beta} \) does not have an abstraction shape). \( \Box \)

Let us now discuss two properties related to **simple typability** for generalized applications, using the original type system of [JM00], which is called here \( ST \). Recall the following typing rules, where \( A, B, C ::= a \mid A \to B \), and \( a \) belongs to a set of base type variables:

\[
\begin{array}{c}
\frac{\Gamma; x : A \vdash t : B}{\Gamma; x : A \vdash x : A} & \frac{\Gamma \vdash \lambda x.t : A \to B}{\Gamma \vdash t : A \to B} & \frac{\Gamma \vdash u : A}{\Gamma; y : B \vdash r : C} & \frac{\Gamma; y : B \vdash r : C}{\Gamma \vdash (u,y,r) : C} \\
\end{array}
\]

We write \( \Gamma \vdash_{ST} t : A \) to denote a type derivation in system \( ST \) ending in the sequent \( \Gamma \vdash t : A \).

**Subformula property.** The subformula property for normal forms is an important property of proof systems, being useful notably for proof search. It holds for von Plato’s generalized natural deduction, and therefore also for the original calculus \( \Delta J \). Despite the minimal amount of permutations used, which does not provide full normal forms, this property is still true in our system.
**Lemma 2.4** (Subformula property). If \( \Phi = \Gamma \vdash_{ST} \text{NF}_{d\beta} : A \) then every formula in the derivation \( \Phi \) is a subformula of \( A \) or a subformula of some formula in \( \Gamma \).

**Proof.** The lemma is proved together with another statement: If \( \Psi = \Gamma \vdash_{ST} \text{NF}_{d\beta} : A \) then every formula in \( \Psi \) is a subformula of some formula in \( \Gamma \). The proof is by simultaneous induction of \( \Phi \) and \( \Psi \).

The subformula property confirms that executing only needed permutations still gives rise to a reasonable notion of normal form.

**Strong Normalization.** The second property for typed terms we show states that they are \( \lambda J_n \)-strongly normalizable. The proof is achieved by mapping \( \lambda J_n \) into the \( \lambda \)-calculus equipped with the following \( \sigma \)-rules [Reg94]:

\[
(\lambda x.M)N N' \mapsto_{\sigma_1} (\lambda x.MN')N \quad (\lambda x.\lambda y.M)N \mapsto_{\sigma_2} \lambda y.(\lambda x.M)N
\]

**Theorem 2.5.** If \( t \) is simply typable, i.e. \( \Gamma \vdash_{ST} t : \sigma \), then \( t \in \text{SN}(d\beta) \).

**Proof.** The proof uses a simple map into the \( \lambda \)-calculus, based on [ES07], given by \( x^\# = x \), \( (\lambda x.t)^\# = \lambda x.t^\# \), and \( (t(u,x.r))^\# = (\lambda x.r^\#)(t^\#_u^\#) \). This map produces the following simulation: if \( t_1 \rightarrow_{d\beta} t_2 \) then \( t_1^\# \rightarrow_{\beta^\sigma_1} t_2^\# \). The proof of the simulation result is by induction on \( t_1 \rightarrow_{d\beta} t_2 \). The base case needs two lemmas: the first one states that map \( (\cdot)^\# \) commutes with substitution; the other, proved by induction on \( d \), states that \( D(\lambda x.t)^\# u^\# \rightarrow_{\beta^\sigma_1} D(\{u/x\}t)^\# \).

Now, given a simply typable term \( t \in T_\beta \), the \( \lambda \)-term \( t^\# \) is also simply typable in the \( \lambda \)-calculus. Hence, \( t^\# \in \text{SN}(\beta, \sigma_1) \). By the simulation result, \( t \in \text{SN}(d\beta) \) follows.

**Confluence.** We now prove confluence of the calculus. For this, we adapt the proof of [Tak95]. The same proof method is used for \( \Lambda J \) by [JM00] and by [ES20] for \( \Lambda J_v \). We begin by defining the following parallel reduction \( \rightarrow_{d\beta} \):

\[
\begin{array}{c}
x \rightarrow_{d\beta} x \\
\lambda x.t \rightarrow_{d\beta} \lambda x.t' \\
t \rightarrow_{d\beta} t' \\
u \rightarrow_{d\beta} u' \\
r \rightarrow_{d\beta} r'
\end{array}
\]

\[
\begin{array}{c}
\text{VAR} \\
\text{ABS} \\
\text{APP} \\
\text{DB}
\end{array}
\]

\[
\begin{array}{c}
D(t) \rightarrow_{d\beta} t' \\
u \rightarrow_{d\beta} u' \\
r \rightarrow_{d\beta} r'
\end{array}
\]

The particularity of our proof is the following lemma which deals with distance.

**Lemma 2.6.** Let \( t_1 = D(t) \rightarrow_{d\beta} t_2 \). Then there are \( D', t' \) such that \( t_2 = D'(t') \) and \( D(\lambda x.t) \rightarrow_{d\beta} D'(\lambda x.t') \).

**Proof.** By induction on \( D \).

**Case:** \( D = \emptyset \). We take \( D' = \emptyset, t' = t_2 \). We have \( \lambda x.t_1 \rightarrow_{d\beta} \lambda x.t_2 \) by rule (ABS).

**Case:** \( D = s(u,y,D_0) \) and \( t_1 = s(u,y,D_0(t)) \rightarrow_{d\beta} s'(u',y,r) = t_2 \) by rule (APP). By hypothesis, we have \( s \rightarrow_{d\beta} s', u \rightarrow_{d\beta} u' \) and \( D_0(t) \rightarrow_{d\beta} r \). By i.h. \( r = D_1(t') \) and \( D_0(\lambda x.t) \rightarrow_{d\beta} D_1(\lambda x.t') \). We conclude by taking \( D' = s'(u',y,D_1) \).
Case: $D = D_0(\lambda z.s)(u, y.D_1)$ and $t_1 = D_0(\lambda z.s)(u, y.D_1(t)) \Rightarrow_{d_\beta} \{\{u'/x\}s'/y\} r = t_2 \text{ by } (\text{ABS}).$ By hypothesis, we have $D_0(\lambda z.s) \Rightarrow_{d_\beta} s', u \Rightarrow_{d_\beta} u'$ and $D_1(t) \Rightarrow_{d_\beta} r$. By i.h. $r = D_2(t''')$ and $D_1(\lambda x.t) \Rightarrow_{d_\beta} D_2(\lambda x.t)'$. We can assume by $\alpha$-equivalence that the free variables of $u'$ and $s'$ are not bound by $D_2$. We take $D' = \{\{u'/z\}s'/y\} D_2$ and $t' = \{\{u'/z\}s'/y\} t'''$. Thus, we have $D'(\lambda x.t') = \{\{u'/z\}s'/y\} D_2(\lambda x.t'')$ and we can conclude $D(\lambda x.t) = D_0(\lambda z.s)(u, y.D_1(\lambda x.t)) \Rightarrow_{d_\beta} D'(\lambda x.t')$ by i.h. and rule (ABS).

Lemma 2.7. Let $y \notin \text{fv}(u)$. Then $\{u/x\} \{r/y\} t = \{u/x\} \{r/y\} \{u/x\} t$.

Proof. Straightforward by induction on $t$. \qed

Lemma 2.8. Let $t_1, t_2, u_1, u_2 \in T_J$. Then:

1. If $t_1 \Rightarrow_{d_\beta} t_2$, then $t_1 \Rightarrow_{d_\beta} t_2$.
2. If $t_1 \Rightarrow^*_{d_\beta} t_2$, then $t_1 \Rightarrow^*_{d_\beta} t_2$.
3. If $t_1 \Rightarrow_{d_\beta} t_2$ and $u_1 \Rightarrow_{d_\beta} u_2$, then $\{u_1/z\} t_1 \Rightarrow_{d_\beta} \{u_2/z\} t_2$.

Proof. The proof of the first statement is by induction on $t_1 \Rightarrow_{d_\beta} t_2$. In the base case $t_1 = D(\lambda x.t)(u, y.r) \Rightarrow_{d_\beta} \{\{u/x\} D(t)/y\} r = t_2$, we use rule (DB) with premises $D(t) \Rightarrow_{d_\beta} D(t)$, $u \Rightarrow_{d_\beta} u$ and $r \Rightarrow_{d_\beta} r$. The other cases are straightforward by i.h. and rules (ABS) or (APP).

The proof of the second statement is by induction on $t_1 \Rightarrow_{d_\beta} t_2$. The base case (VAR) is by an empty reduction $t_1 = x = t_2$. The cases (ABS) and (APP) are direct by i.h. The case left is (DB), with $t_1 = D(\lambda x.t)(u, y.r) \Rightarrow_{d_\beta} \{\{u/x\} t'/y\} r' = t_2$ with hypothesis $D(t) \Rightarrow_{d_\beta} t'$, $D(u) \Rightarrow_{d_\beta} u'$ and $D(r) \Rightarrow_{d_\beta} r'$. By Lemma 2.6, there are $D', t''$ such that $D(\lambda x.t) \Rightarrow_{d_\beta} D'(\lambda x.t'')$ and $t' = D'(t'')$. By i.h. we have $D(\lambda x.t) \Rightarrow^*_{d_\beta} D'(\lambda x.t'')$, $u \Rightarrow_{d_\beta} u'$ and $r \Rightarrow_{d_\beta} r'$. We have the following reduction:

$t_1 \Rightarrow^*_{d_\beta} D'(\lambda x.t'')(u', y.r') \Rightarrow_{d_\beta} \{\{u'/x\} D'(t'')/y\} r' = t_2$.

The proof of the third statement is also by induction on $t_1 \Rightarrow_{d_\beta} t_2$.

Case (VAR): Then $t_1$ is a variable. If $t_1 = z$, we have $\{u_1/z\} t_1 = u_1$, $\{u_2/z\} t_2 = u_2$ and this is direct by the second hypothesis. If $t_1 = y \neq z$, we have $\{u_1/z\} t_1 = y = \{u_2/z\} t_2$, this is direct by (VAR).

Case (ABS): Then $t_1 = \lambda x.t \Rightarrow_{d_\beta} \lambda x.t' = t_2$, where w.l.o.g. $x \neq z$ and $x \notin \text{fv}(u_1) \cup \text{fv}(u_2)$ and such that $t \Rightarrow_{d_\beta} t'$ by i.h. we have $\{u_1/z\} t_1 = \lambda x.\{u_1/z\} t = \lambda x.\{u_2/z\} t' = \{u_2/z\} t_2$.

Case (APP): Then $t_1 = t(u, x.r) \Rightarrow_{d_\beta} t'(u, x.r') = t_2$, where w.l.o.g. $x \neq z$ and $x \notin \text{fv}(u_1) \cup \text{fv}(u_2)$ and such that $t \Rightarrow_{d_\beta} t'$ by i.h. we have $\{u_1/z\} t_1 = \{u_1/z\} t(u, x.\{u_1/z\} r) = \{u_1/z\} u.\{u_1/z\} t' = \{u_1/z\} \{u_2/z\} u' = \{u_2/z\} t_2$.

Case (DB): Then $t_1 = D(\lambda x.t)(u, y.r) \Rightarrow_{d_\beta} \{\{u/x\} t'/y\} r = t_2$ where w.l.o.g $x, y \neq z$ and $x, y \notin \text{fv}(u_1) \cup \text{fv}(u_2)$, $D$ does not capture free variables of $u_1, u_2$, and such that $D(t) \Rightarrow_{d_\beta} t'$, $u \Rightarrow_{d_\beta} u'$ and $r \Rightarrow_{d_\beta} r'$. By i.h. we have $\{u_1/z\} D(t) \Rightarrow_{d_\beta} \{u_2/z\} t'$, $\{u_1/z\} u \Rightarrow_{d_\beta} \{u_2/z\} u'$ and $\{u_1/z\} r \Rightarrow_{d_\beta} \{u_2/z\} r'$. Let $\{u_1/z\} D(t) = D_{\{u_1/z\}}(t_{\{u_1/z\}}).$

By rule (DB), we infer

$\{u_1/z\} t_1 = D_{\{u_1/z\}}(\lambda x.t_{\{u_1/z\}})\{\{u_1/z\} u, y, \{u_1/z\} r)$

$\Rightarrow_{d_\beta} \{\{u_2/z\} u'/x\} \{u_2/z\} t'/y\} \{u_2/z\} r$

$= \{u_2/z\} t_2$ (by Lemma 2.7 twice) \qed
Statements (1) and (2) of the previous lemma imply that $\to_{\beta J}^*$ is the transitive and reflexive closure of $\equiv_{\beta J}$. We now only need to prove the diamond property for $\equiv_{\beta J}$ to conclude. The difference between Takahashi’s method and the more usual Tait and Martin-Löf’s method [Bar84, §3.2] is to replace the proof of diamond for the parallel reduction by a proof of the triangle property.

**Definition 2.9** (Triangle property). Let $\to_{\beta J}$ be a reduction relation on $T_J$ and $f$ a function. $(\to_{\beta J}, f)$ satisfies the triangle property if, for any $t \in T_J$, $t \to_{\beta J} t'$ implies $t' \to_{\beta J} f(t)$.

**Definition 2.10** (Developments). The $d_{\beta}$-development $(t)^{d_{\beta}}$ of a $T_J$-term $t$ is defined as follows.

$$(x)^{d_{\beta}} = x \quad (\lambda x.t)^{d_{\beta}} = \lambda x. (t)^{d_{\beta}} \quad (t(u, y, r))^ {d_{\beta}} = \{\{(u)^{d_{\beta}} / x\} (D(t))^ {d_{\beta}} / y\} (r)^ {d_{\beta}}, \text{ if } t = D(\lambda x.t')$$

$$(t)^{d_{\beta}} (u)^ {d_{\beta}} = (t, x)^{d_{\beta}}$$

**Lemma 2.11** (Triangle property of $(\equiv_{d_{\beta}}, (\cdot)^{d_{\beta}})$). Let $t_1 \equiv_{d_{\beta}} t_2$. Then $t_2 \equiv_{d_{\beta}} (t_1)^{d_{\beta}}$.

**Proof.** By induction on $t_1$.

Case $t_1 = x$: Then $t_1 = t_2 = (t_1)^{d_{\beta}}$ and we conclude with rule (VAR).

Case $t_1 = \lambda x. t$: Then $t_1 \equiv_{d_{\beta}} t_2 = \lambda x. t'$ by rule (ABS). We have $(t_1)^{d_{\beta}} = \lambda x. (t)^{d_{\beta}}$. By i.h. $t' \equiv_{d_{\beta}} (t)^{d_{\beta}}$. By (ABS), $\lambda x. t' \equiv_{d_{\beta}} \lambda x. (t)^{d_{\beta}}$.

Case $t_1 = t(u, y, r)$, where $t \neq D(\lambda x. t')$: Then $t_1 \equiv_{d_{\beta}} t_2 = t'(u, y, r')$ by rule (APP). We have $(t_1)^{d_{\beta}} = (t)^{d_{\beta}} (u)^{d_{\beta}}, y. (r)^{d_{\beta}}$. By i.h. $t' \equiv_{d_{\beta}} (t)^{d_{\beta}}$, $u' \equiv_{d_{\beta}} (u)^{d_{\beta}}$, and $r' \equiv_{d_{\beta}} (r)^{d_{\beta}}$. By (APP), $t'(u', y, r') \equiv_{d_{\beta}} (t_1)^{d_{\beta}}$.

Case $t_1 = D(\lambda x. t)(u, y, r)$: Then $(t_1)^{d_{\beta}} = \{(u)^{d_{\beta}} / x\} (D(t))^ {d_{\beta}} / y\} (r)^ {d_{\beta}}$. There are two sub-cases. In both cases we have $u \equiv_{d_{\beta}} u'$ and $r \equiv_{d_{\beta}} r'$ and thus by i.h. $u' \equiv_{d_{\beta}} (u)^{d_{\beta}}$ and $r \equiv_{d_{\beta}} (r)^{d_{\beta}}$.

Subcase (APP): Then $D(\lambda x. t) \equiv_{d_{\beta}} t'$. By a reasoning similar to Lemma 2.6, we can show that $t' = D'(\lambda x. t')$ and that $D(t) \equiv_{d_{\beta}} D'(t')$. Thus $t_2 = D'(\lambda x. t')(u', y, r')$ and by i.h. $D'(t') \equiv_{d_{\beta}} (D(t))^ {d_{\beta}}$. We use rule (DB) with the three i.h. as premises to derive $t_2 \equiv_{d_{\beta}} \{(u)^{d_{\beta}} / x\} (D(t))^ {d_{\beta}} / y\} (r)^ {d_{\beta}} = (t_1)^{d_{\beta}}$.

Subcase (DB): Then $t_2 = \{(u'/x) / y\} t' / y\) r'$. By i.h. $t' \equiv_{d_{\beta}} (D(t))^ {d_{\beta}}$. By i.h. and two applications of Lemma 2.8(3), we have:

$$\{(u'/x) / y\} t' / y\) r' \equiv_{d_{\beta}} \{(u)^{d_{\beta}} / x\} (D(t))^ {d_{\beta}} / y\} (r)^ {d_{\beta}} = (t_1)^{d_{\beta}}$$

**Proposition 2.12.** The reduction relation $\to_{d_{\beta}}$ is confluent.

**Proof.** The triangle property of $(\equiv_{d_{\beta}}, (\cdot)^{d_{\beta}})$ implies that $\equiv_{d_{\beta}}$ is diamond, since for any $t_2$ such that $t_1 \equiv_{d_{\beta}} t_2$, we have $t_2 \equiv_{d_{\beta}} (t_1)^{d_{\beta}}$. This implies in turn that $\equiv_{d_{\beta}} = \to_{d_{\beta}}^*$ is diamond and thus that $\to_{d_{\beta}}$ is confluent.

3. **Inductive Characterization of Strong Normalization**

In this section we give an inductive characterization of strong normalization (ISN) for $\lambda J_n$, written $\text{ISN}(d_{\beta})$, and prove it correct. This characterization will be useful to show completeness of the type system that we are going to present in subsection 4.1, as well as to compare strong normalization of $\lambda J_n$ to the ones of $T_J[\beta, p2]$ and $\Lambda J$.
3.1. **ISN in the λ-calculus with Weak-Head Contexts.** We write ISN(\(R\)) the set of strongly normalizing terms under \(R\) given by an inductive definition. As an introduction, we first look at the case of ISN for the λ-calculus (written ISN(\(\beta\))), on which our forthcoming definition of ISN(\(d\beta\)) elaborates. A usual way to define ISN(\(\beta\)) is by the following rules [vR96], where the general notation \(\vec{P}\) abbreviates \((\ldots (MP_1) \ldots)P_n\) for some \(n \geq 0\).

\[
\begin{align*}
\text{P}_1, \ldots, P_n &\in \text{ISN}(\beta) & M &\in \text{ISN}(\beta) & \{N/x\}M\vec{P}, N &\in \text{ISN}(\beta) \\
x\vec{P} &\in \text{ISN}(\beta) & \lambda x. M &\in \text{ISN}(\beta) & (\lambda x. M)N\vec{P} &\in \text{ISN}(\beta)
\end{align*}
\]

One then shows that \(M \in SN(\beta)\) if and only if \(M \in \text{ISN}(\beta)\).

Notice that this definition is deterministic. Indeed, a reduction strategy emerges from this definition: it is a strong strategy based on a preliminary weak-head strategy. The strategy is the following: first reduce a term to a weak-head normal form \(\lambda x. M\) or \(x\vec{P}\), and then iterate reduction under abstractions and inside arguments (in any order), without any need to come back to the head of the term. Formally, weak-head normal forms, which are those produced by the first level of the strategy, are of two kinds:

- **Neutral terms** \(n := x \mid nM\)
- **Answers** \(a := \lambda x. M\)

Neutral terms cannot produce any head \(\beta\)-redex. They are the terms of the shape \(x\vec{P}\). On the contrary, answers can create a \(\beta\)-redex when given at least one argument. In the case of the \(\lambda\)-calculus, these are only abstractions. If the term is not a weak-head normal form, a redex can be located inside a

- **(Weak-head context)** \(W := \diamond \mid Wt\).

These concepts give rise to a different definition of ISN(\(\beta\)):

\[
\begin{align*}
x &\in \text{ISN}(\beta) & n.M &\in \text{ISN}(\beta) & M &\in \text{ISN}(\beta) & W(\{N/x\}M), N &\in \text{ISN}(\beta) \\
nM &\in \text{ISN}(\beta) & \lambda x. M &\in \text{ISN}(\beta) & W((\lambda x. M)N) &\in \text{ISN}(\beta)
\end{align*}
\]

Weak-head contexts are an alternative to the meta-syntactic notation \(\vec{P}\) of vectors of arguments used in the first definition of ISN(\(\beta\)). Notice in the alternative definition 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 d\(\beta\).** We now define ISN(\(d\beta\)) with the same tools used in the last subsection. Hence, we first have to define neutral terms, answers and a notion of contexts. We call the contexts left-right contexts (\(R\)), and the underlying strategy the left-right strategy.

**Definition 3.1.** We consider the following grammars:

- **(Neutral terms)** \(n := x \mid n(u, x.n)\)
- **(Answers)** \(a := \lambda x. t \mid n(u, x.a)\)
- **(Left-right contexts)** \(R := \diamond \mid R(u, x.r) \mid n(u, x.R)\)

Notice that \(n\) and \(a\) are disjoint and stable by \(d\beta\)-reduction. Notice also that this time, answers are not only abstractions, but also abstractions under a special distant context. Moreover \(n(u, x.r)\) is never a \(d\beta\)-redex, whereas \(a(u, x.r)\) is always a \(d\beta\)-redex. The terminology “left-right” intends to suggest that the hole \(\diamond\) may appear in the left (viz \(R(u, x.r)\)) or right (viz \(n(u, x.R)\)) component of generalized applications. If this last form of \(R\) was
forbidden, then we would define the contexts by \( \mathcal{W} := \diamond | \mathcal{W}(u, x.r) \), a generalized form of weak-head contexts from the \( \lambda \)-calculus, actually implicitly used in [Mat00] for \( \Lambda J \) (see also Fig. 1 in Section 6). However, these contexts \( \mathcal{W} \) are not convenient for defining an inductive predicate of strong normalization based on the distant rule \( d_\beta \), as shown below in Remark 3.6.

To achieve a characterization of \( \text{ISN}(d_\beta) \), we still need to obtain a deterministic notion of decomposition, that we explain by means of an example.

**Example 3.2 (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 \( d_\beta \)-redex \( r \) and a left-right context \( R \), *i.e.* there are two ways to write \( t \) as \( R(r) \): either \( R = \emptyset \) and \( r = t = D(I)(x_3, y.II) \), for some \( D \); or \( R = x_1(x_2, y_1.\emptyset)(x_3.y.II) \) and \( r = I(I, z.I) \). Notice how in the second case all the three rules in the grammar of left-right contexts are needed to generate \( R \).

In the previous example, we will rule out the first decomposition by defining next a restriction of the \( d_\beta \)-rule, securing uniqueness of such kind of decomposition in all cases. For that, we introduce a restricted notion of distant context:

\[
(\text{Neutral distant contexts}) \quad D_n := \emptyset | n(u, x.D_n)
\]

Notice \( D_n \subseteq R \); moreover, \( D_n(\lambda x.t) \) is an answer \( a \) and conversely every answer has that form.

The reduction relation underlying our definition of \( \text{ISN}(d_\beta) \) is the left-right reduction \( \rightarrow_{\text{lr}} \), defined as the closure under \( R \) of the following **restricted** \( d_\beta \)-rule:

\[
D_n(\lambda x.t)(u, y.r) \mapsto \{D_n(\{u/y\}t)/y\}r.
\]

The restriction of \( D \) to a neutral distant context \( D_n \) is what allows determinism of our reduction relation \( \rightarrow_{\text{lr}} \) (Lemma 3.5) and correctness of our forthcoming definition of \( \text{ISN}(d_\beta) \) (Definition 3.7).

**Example 3.3 (Decomposition).** Going back to Example 3.2, did we obtain a decomposition \( R(r) \) for \( t \), with \( r \) a restricted \( d_\beta \)-redex? The first option fails because \( D \) is not a neutral distant context; and the second option succeeds because \( I(I, z.I) \) is of course a restricted redex.

Coherently with the \( \lambda \)-calculus, left-right normal forms are either neutral terms or answers.

**Lemma 3.4.** Let \( t \in T_J \). Then \( t \) is in lr-normal form iff \( t \in n \cup a \).

**Proof.** First, we show that \( t \) lr-normal implies \( t \in n \cup a \), by induction on \( t \). If \( t = x \), then \( t \in n \). If \( t = \lambda x.s \), then \( t \in a \). Let \( t = s(u, x.r) \) where \( s \) and \( r \) are lr-normal. Then \( s \notin a \), otherwise the term would lr-reduce at root. Thus by the i.h. \( s \in n \). By the i.h. again \( r \in n \cup a \) so that \( t \in n \cup a \).

Second, we show that \( t \in n \cup a \) implies \( t \) is lr-normal, by simultaneous induction on \( n \) and \( a \). The cases \( t = x \) (i.e. \( t \in n \)) and \( t = \lambda x.s \) (i.e. \( t \in a \)) are straightforward. Let \( t = s(u, x.r) \) where \( s \in n \) and \( r \in n \cup a \). Since \( r, s \in n \cup a \), by the i.h. \( t \) does not lr-reduce in \( r \) or \( s \). Since \( s \in n \), \( t \) does not lr-reduce at root either. Then, \( t \) is lr-normal.

Moreover, determinism of \( \rightarrow_{\text{lr}} \) holds.

**Lemma 3.5.** The reduction \( \rightarrow_{\text{lr}} \) is deterministic.
Proof. Let \( t \) be a lr-reducible term. We reason by induction on \( t \). If \( t \) is a variable or an abstraction, then \( t \) does not lr-reduce so that \( t \) is necessarily an application \( t'(u, y. r) \). By Lemma 3.4 we have three possible cases for \( t' \).

**Case** \( t = t'(u, y. r) \) with \( t' \in a \): Then \( t = D_n(\lambda x. s)(u, y. r) \), so \( t \) reduces at the root. Since \( t' \in a \), then we know by Lemma 3.4 that (1) \( t' \in \text{NF}_{lr} \), (2) \( t' \notin n \), so that \( t \) does not lr-reduce in \( t' \) or \( r \).

**Case** \( t = t'(u, y. r) \) with \( t' \in n \): Then \( t \) does not lr-reduce at the root. By Lemma 3.4, we know that \( t' \in \text{NF}_{lr} \) and thus \( t \) necessarily reduces in \( r \). By the i.h. this reduction is deterministic.

**Case** \( t = t'(u, y. r) \) with \( t' \notin \text{NF}_l \): Then in particular by Lemma 3.4 we know that (1) \( t' \) does not have an abstraction shape so that \( t \) does not reduce at the root, and (2) \( t' \notin n \) so that \( t \) does not reduce in \( r \). Thus \( t \) lr-reduces only in \( t' \). By the i.h. this reduction is deterministic.

**Remark 3.6.** Consider again the term \( t = x_1(x_2, y_1. I(I, z. I))(x_3, y. \Pi) \) in Example 3.2. As we explained before, if the form \( n(u, x. r) \) of the grammar of \( R \) was disallowed, then it would not be possible to decompose \( t \) as \( R(r) \), with \( r \) a restricted \( \beta \)-redex. Moreover, the reduction strategy associated with the intended definition of ISN(\( d \beta \)) would consider \( t \) as a left-right normal form, and start reducing the subterms of \( t \), including \( I(I, z. I) \).

Now, this latter (internal) subterm would eventually reach \( I \) and suddenly the whole term \( t' = x_1(x_2, y_1. I)(x_3, y. r') \) would become an external left-right redex: the typical separation between an initial external reduction phase followed by an internal reduction phase —as it is the case in the \( \lambda \)-calculus— would be lost in our framework. This point due to the distant character of rule \( d \beta \) explains the subtlety of Definition 3.7.

Our inductive definition of strong normalization follows.

**Definition 3.7 (Inductive strong normalization).** We consider the following inductive predicate:

\[
\begin{align*}
\text{SNVAR:} & \quad x \in \text{ISN}(d \beta) \\
\text{SNAPP:} & \quad n, u, r \in \text{ISN}(d \beta) \quad r \in \text{NF}_{lr} \\
\text{SNABS:} & \quad \lambda x.t \in \text{ISN}(d \beta) \\
\text{SNBETA:} & \quad R\langle D_n(\{u/x\}t)/y. r), D_n(t), u \in \text{ISN}(d \beta) \rangle \in \text{ISN}(d \beta)
\end{align*}
\]

Notice that every term can be written according to the conclusions of the previous rules, so that the following grammar also defines the syntax \( T_J \).

\[
t, u, r ::= x \mid \lambda x.t \mid n(u, x. \text{NF}_{lr}) \mid R(D_n(\lambda x.t)(u, y. r)) \tag{3.1}
\]

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 "\( r \in \text{NF}_{lr} \)" in rule (SNAPP). Indeed, this (weaker) rule would overlap with rule (SNBETA) for terms in which the left-right context lies in the last continuation, as for instance in \( x(u, y. y)(u', y'. \Pi) \). Notice the difference with the \( \lambda \)-calculus: due to the definition of left-right contexts \( R \), the head of a term with generalized applications can be either on the left of the term (as in the \( \lambda \)-calculus), or recursively on the left in a continuation.

To show that our definition corresponds to strong normalization, we need a few intermediate statements.
Lemma 3.8. If $t_0 \rightarrow_{d\beta} t_1$, then
- $\{u/x\}t_0 \rightarrow_{d\beta} \{u/x\}t_1$, and
- $\{t_0/x\}u \rightarrow_{d\beta}^{*} \{t_1/x\}u$.

Proof. In the base cases, we have $t_0 = D(\lambda z.t)(s, y, r) \rightarrow_{d\beta} \{\{s/z\}D(t)/y\}r = t_1$. By $\alpha$-equivalence we can suppose that $y, z \notin \mathbb{F}(u)$ and $x \neq y, x \neq z$. The inductive cases and the base case for item (2) are straightforward. We detail the base case of item (1).

$$
\{u/x\}t_0 = \{u/x\} D(\lambda z.t)(\{u/x\}s, y, \{u/x\}r)
\rightarrow_{d\beta} \{\{u/x\}s/z\} \{u/x\} D(t)/y \{u/x\}r
= \text{Lemma } 2.7 \{\{u/x\} (\{s/z\} D(t))/y\} \{u/x\}r
= \text{Lemma } 2.7 \{u/x\} \{\{s/z\} D(t)/y\}r
= \{u/x\} t_1
$$

□

Remark 3.9. For any $T_{\beta}$-term $D(\lambda x.t) \in SN(d\beta) \iff D(t) \in SN(d\beta)$.

Lemma 3.10. Let $t_0 = R(\{\{u/x\} D(t)/y\} r), D(t), u \in SN(d\beta)$. Then $t_0' = R(D(\lambda x.t)(u, y, r)) \in SN(d\beta)$.

Proof. In this proof we use a notion of reduction of contexts which is the expected one: $C \rightarrow C'$ iff the hole in $C$ is outside the redex contracted in the reduction step. By hypothesis we also have $r \in SN(d\beta)$. We use the lexicographic order to reason by induction on $(||t_0||_{d\beta}, ||D(t)||_{d\beta}, ||u||_{d\beta})$. To show $t_0' \in SN(d\beta)$ it is sufficient to show that all its reducts are in $SN(d\beta)$. We analyze all possible cases.

Case $t_0' \rightarrow_{d\beta} t_0$: We conclude by the hypothesis.

Case $t_0' \rightarrow_{d\beta} R(D(\lambda x.t')(u, y, r)) = t_1'$, where $t \rightarrow_{d\beta} t'$: Thus also $D(t) \rightarrow_{d\beta} D(t')$. We then have $D(t') \in SN(d\beta)$ and $u \in SN(d\beta)$ and by item (2) we have $t_0 = R(\{\{u/x\} D(t)/y\} r) \rightarrow_{d\beta}^{*} R(\{\{u/x\} D(t')/y\} r) = t_1'$, so that also $t_1 \in SN(d\beta)$. We can conclude that $t_1' \in SN(d\beta)$ by the i.h. since $||t_1||_{d\beta} \leq ||t_0||_{d\beta}$ and $||D(t')||_{d\beta} < ||D(t)||_{d\beta}$.

Case $t_0' \rightarrow_{d\beta} R(D(\lambda x.t)(u', y, r)) = t_1'$, where $u \rightarrow_{d\beta} u'$: We have $D(t), u \in SN(d\beta)$ and by item (2) $t_0 = R(\{\{u/x\} D(t)/y\} r) \rightarrow_{d\beta}^{*} R(\{\{u'/x\} D(t)/y\} r) = t_1'$, so that also $t_1 \in SN(d\beta)$. We conclude $t_1' \in SN(d\beta)$ by the i.h. since $||t_1||_{d\beta} < ||t_0||_{d\beta}$ and $||u'||_{d\beta} < ||u||_{d\beta}$.

Case $t_0' \rightarrow_{d\beta} R(D(\lambda x.t)(u, y', r')) = t_1'$, where $r \rightarrow_{d\beta} r'$: We have $D(t), u \in SN(d\beta)$ and by item (1) $t_0 = R(\{\{u/x\} D(t)/y\} r) \rightarrow_{d\beta}^{*} R(\{\{u/x\} D(t)/y\} r') = t_1'$. We conclude $t_1' \in SN(d\beta)$ by the i.h. since $||t_1||_{d\beta} < ||t_0||_{d\beta}$.

Case $t_0' \rightarrow_{d\beta} R'(D(\lambda x.t)(u, y, r')) = t_1'$, where $D \rightarrow_{d\beta} D'$: We have $D'(t), u \in SN(d\beta)$ and by Lemma 3.8 $t_0 = R(\{\{u/x\} D(t)/y\} r) \rightarrow_{d\beta}^{*} R(\{\{u/x\} D'(t)/y\} r) = t_1'$, so that also $t_1 \in SN(d\beta)$. We conclude $t_1' \in SN(d\beta)$ by the i.h. since $||t_1||_{d\beta} < ||t_0||_{d\beta}$ and $||D'(t)||_{d\beta} < ||D(t)||_{d\beta}$.

Case $t_0' \rightarrow_{d\beta} R'(D(\lambda x.t)(u, y, r)) = t_1'$, where $R \rightarrow_{d\beta} R'$: Thus $t_0 = R(\{\{u/x\} D(t)/y\} r) \rightarrow_{d\beta}^{*} R'(\{\{u/x\} D(t)/y\} r) = t_1'$. We have $t_1, D(t), u \in SN(d\beta)$. We conclude that $t_1' \in SN(d\beta)$ by the i.h. since $||t_1||_{d\beta} < ||t_0||_{d\beta}$.

Case $R = R'(D_0(u, y, r'))$ and $r = D''(\lambda x'.t')$: This is the only case left. Indeed, there is no redex in $D(\lambda x.t)$ other than in $D$ or $\lambda x.t$. Then,

$$
t_0' = R'(D_0(D(\lambda x.t)(u, y, D''(\lambda x'.t')))(u', y', r'))
$$
Let $D' = D_n(D_n(\lambda x.t)(u,y,D''))$. The reduction we need to consider is:

\[
t'_0 = R'(D'(\lambda x'.t')(u',y'.r')) \\
\rightarrow_{d_β} R'((\{u'/x'\})D'(t'/y')r') \\
= R'((\{u'/x'\})D_n(D_n(\lambda x.t)(u,y,D''(t'))/y')r') = t'_1
\]

We will show that $t'_1 \in SN(d_β)$.

For this we show that $t_1 = R'((\{u'/x'\})D_n((\{u/x\}D(t)/y)D''(t')/y')r') \in SN(d_β)$, that $D'(t') \in SN(d_β)$ and that $u' \in SN(d_β)$. We have $t_0 \rightarrow_{d_β} t_1$ so that $t_1 \in SN(d_β)$ and $||t_1||_{d_β} < ||t_0||_{d_β}$. $u'$ is a subterm of $t_0$, which is in SN(d_β), so that $u' \in SN(d_β)$. To show that $D'(t') \in SN(d_β)$, we consider $t_2 = D_n((\{u/x\}D(t)/y)D''(\lambda x'.t'))$. We have $t_0 = R'(t_2(u',y'.r'))$. We can show that $||t_2||_{d_β} < ||t_0||_{d_β}$ (so that $t_2 \in SN(d_β)$). Indeed, $|R'(t_2(u',y'.r'))|_{d_β} \geq |t_2(u',y'.r')|_{d_β} \geq ||t_2||_{d_β} + 1 > ||t_0||_{d_β}$. The second inequality holds since $t_2$ has an abstraction shape, and abstraction shapes are stable under substitution, and thus $t_2(u',y'.r')$ is also a redex. We can then conclude that $t'_2 = D_n(D_n(\lambda x.t)(u,y,D''(\lambda x'.t'))) = D'(\lambda x'.t') \in SN(d_β)$ by the i.h. since $u, D(t) \in SN(d_β)$. Thus, $D'(t') \in SN(d_β)$ by Remark 3.9.

We then have $t_1, D'(t'), u' \in SN(d_β)$ and we can conclude $t'_1 \in SN(d_β)$ since $||t_1||_{d_β} < ||t_0||_{d_β}$. We conclude $t'_1 \in SN(d_β)$ as required.

**Theorem 3.11.** $SN(d_β) = ISN(d_β)$.

**Proof.** First, we show $ISN(d_β) \subseteq SN(d_β)$. We proceed by induction on $t \in ISN(d_β)$.

**Case t = x:** Straightforward.

**Case t = λx.s,** where $s \in ISN(d_β)$: By the i.h. $s \in SN(d_β)$, so that $t \in SN(d_β)$ trivially holds.

**Case t = s(u, r) \in NF_h** where $s, u, r \in ISN(d_β)$: By Lemma 3.4 we have $s \in n$ and thus in particular $s$ can not $d_β$-reduce to an answer. Therefore any kind of reduction starting at $t$ only occurs in the subterms $s$, $u$ and $r$. We conclude since by the i.h. we have $s, u, r \in SN(d_β)$.

**Case t = R(D_n(\lambda x.s)(u, y.r))**, where $R((\{u/x\}D_n(s)/y)r), D_n(s), u \in ISN(d_β)$: The i.h. gives $R((\{u/x\}D_n(s)/y)r) \in SN(d_β), D_n(s) \in SN(d_β)$ and $u \in SN(d_β)$ so that by Lemma 3.10 $t = R(D_n(\lambda x.s)(u, y.r)) \in SN(d_β)$ holds, with $D = D_n$.

Next, we show $SN(d_β) \subseteq ISN(d_β)$. Let $t \in SN(d_β)$. We reason by induction on $||t||_{d_β}$ w.r.t. the lexicographic order. If $||t||_{d_β} = 0$, then $t$ is a variable and thus in ISN(d_β) by rule (SNVAR). Otherwise we proceed by case analysis.

**Case t = λx.s:** Since $||s||_{d_β} \leq ||t||_{d_β}$ and $|s| < |t|$, we conclude by the i.h. and rule (SNABS).

**Case t is an application:** There are two cases.

**Subcase t ∈ NF_h:** Then $t = s(u, x.r)$ with $s, u, r \in SN(d_β)$ and $s \in n$. We have $||s||_{d_β} \leq ||t||_{d_β}$, $||u||_{d_β} \leq ||t||_{d_β}$, $||r||_{d_β} \leq ||t||_{d_β}$, $|s| < |t|$, $|u| < |t|$ and $|r| < |t|$. By the i.h. $s, u, r \in ISN(d_β)$ and thus we conclude by rule (SNAPP).

**Subcase t ∉ NF_h:** By definition there is a context $R$ s.t. $t = R(D_n(\lambda x.s)(u, y.r))$. Moreover, $t \in SN(d_β)$ implies in particular $R((\{u/x\}D_n(s)/y)r), u \in SN(d_β)$, so that they are in ISN(d_β) by the i.h. Moreover, $t \in SN(d_β)$ also implies $D_n(\lambda x.s) \in SN(d_β)$. Since the abstraction $\lambda x.s$ is never applied nor an argument, this is equivalent to $D_n(s) \in SN(d_β)$, thus $D_n(s) \in ISN(d_β)$ by the i.h. We conclude by rule (SNBETA).
4. Quantitative Types Capture Strong Normalization

We proved in subsection 2.3 that simply typable terms are strongly normalizing. In this section we use non-idempotent intersection types to fully characterize strong normalization, so that not only typable terms are strongly normalizing, but also strongly normalizing terms are typable. First we introduce the typing system, next we prove the characterization, and finally we study the quantitative behavior of the permutative rule $\pi$ by giving in particular an example of failure of type preservation along $\pi$.

4.1. The Typing System. We define the quantitative type system $\cap J$ for $T_J$-terms and we show that strong normalization in $\lambda J$ 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 and multiset types:

\[
\begin{align*}
\text{(Types)} & \quad \sigma, \tau, \rho \ ::= \ a \mid M \to \sigma \\
\text{(Multiset types)} & \quad M, N \ ::= [\sigma]_{i \in I} \text{ where } I \text{ is a finite set}
\end{align*}
\]

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

Typing environments (or just environments), written $\Gamma, \Delta, \Lambda$, are functions from variables to multiset types assigning the empty multiset to all but a finite set of variables. The domain of $\Gamma$ is given by $\text{dom}(\Gamma) := \{x \mid \Gamma(x) \neq []\}$. The union of environments, written $\Gamma \sqcup \Delta$, is defined by $(\Gamma \sqcup \Delta)(x) := \Gamma(x) \sqcup \Delta(x)$, where $\sqcup$ denotes multiset union. This notion is extended to several environments as expected, so that $\forall i \in I \Gamma_i$ denotes a finite union of environments ($\forall i \in I \Gamma_i$ is to be understood as the empty environment when $I = \emptyset$).

We write $\Gamma \setminus x$ for the environment such that $(\Gamma \setminus x)(y) = \Gamma(y)$ if $y \neq x$ and $(\Gamma \setminus x)(x) = []$. We write $\Gamma ; \Delta$ for $\Gamma \sqcup \Delta$ when $\text{dom}(\Gamma) \cap \text{dom}(\Delta) = \emptyset$. A sequent has the form $\Gamma \vdash t : \sigma$ or $\Gamma \vdash t : M$, where $\Gamma$ is an environment, $t$ is a term, $\sigma$ is a type and $M$ a multiset type.

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

\[
\begin{align*}
\Gamma ; x : \sigma \vdash x : \sigma \quad \text{(VAR)} & \quad \Gamma \vdash \lambda x. t : M 
\to \sigma \quad \text{(ABS)} & \quad \frac{\forall i \in I \Gamma_i \vdash t : [\sigma]_{i \in I} \quad I \neq \emptyset}{\Gamma \vdash t : [\sigma]_{i \in I} \quad \text{(MANY)}}
\end{align*}
\]

\[
\frac{\Gamma \vdash t : \#([M_i \to \tau_i]_{i \in I}) \quad \Delta \vdash u : \#(\sqcup_{i \in I} M_i)}{\Gamma \sqcup \Delta \sqcup \Lambda \vdash t(u, x, r) : \sigma \quad \text{(APP)}}
\]

The typing system handles sequents assigning a type $\sigma$ or a multiset $[\sigma]_{i \in I}$, with $I \neq \emptyset$. According to the rule (MANY), the latter kind of sequents should be understood as a shorthand for a set of sequents of the former kind. Still, the case $I = \emptyset$ is possible in rule (APP), this is precisely when the subtle use of the choice operator is required. Indeed, if $I$ is empty in (APP), meaning in particular that $x : []$ appears in the typing environment of the third premise, then the multisets $[M_i \to \tau_i]_{i \in I}$ and $\sqcup_{i \in I} M_i$ are both empty. Therefore, the choice operator must be used to type both terms $t$ and $u$, which cannot be assigned the empty multiset type. In this case, the resulting types $\#([M_i \to \tau_i]_{i \in I})$ and $\#(\sqcup_{i \in I} M_i)$ are non-empty multiset types, but they are not necessarily related (c.f. forthcoming example.)
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 $[\varepsilon] \rightarrow \sigma$ types $t$.

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 (APP), then, as mentioned before, the types of $t$ and $u$ are not necessarily related. Indeed, let $t := \delta(x, y, x, y)$, then $t$ is $d\beta$-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 (see Lemma 4.1), then the unrelated types $(\mathcal{M}_i \rightarrow \tau_i)_{i \in I}$ of the two occurrences of $\delta$ witness the fact that these subterms will never interact during the reduction of $t$. Indeed, the term $t$ can be typed as follows, where $\rho_i := [[\sigma_i] \rightarrow \sigma_i, \sigma_i] \rightarrow \sigma_i$ and $\tau_i := [\sigma_i] \rightarrow \sigma_i$, for $i = 1, 2$:

\[
\begin{align*}
\emptyset \vdash \delta : \rho_1 &\quad \text{(MANY)} \\
\emptyset \vdash \delta : [\rho_1] &\quad \text{(MANY)} \\
z : [\tau] \vdash \delta(x, y, x, y) : \tau &\quad \text{(APP)}
\end{align*}
\]

where $\delta$ is typed with $\rho_i$ as follows:

\[
\begin{align*}
y : [\tau_i] \vdash y : [\hat{\tau}] &\quad \text{(VAR)} \\
y : [\hat{\tau}] \vdash y : [\hat{\tau}] &\quad \text{(VAR)} \\
y : [\sigma_i] \vdash y : [\hat{\sigma_i}] &\quad \text{(VAR)} \\
y : [\sigma_i] \vdash y : [\sigma_i] &\quad \text{(VAR)} \\
y : [[\sigma_i] \rightarrow \sigma_i, \sigma_i] \vdash y(y, w, w) : \sigma_i &\quad \text{(APP)} \\
\emptyset \vdash \lambda y. y(y, w, w) : [[\sigma_i] \rightarrow \sigma_i, \sigma_i] \rightarrow \sigma_i &\quad \text{(ABS)}
\end{align*}
\]

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

**Lemma 4.1** (Relevance). If $\Gamma \vdash t : \sigma$, then $fv(t) = dom(\Gamma)$.

**Lemma 4.2** (Split).

- If $\Gamma \vdash^n t : \mathcal{M}$, then for any decomposition $\mathcal{M} = \sqcup_{i \in I} \mathcal{M}_i$, where $\mathcal{M}_i \neq \emptyset$ for all $i \in I$, then we have $\Gamma_n \vdash^n t : \mathcal{M}$ such that $\sum_{i \in I} n_i = n$ and $\omega_i \in I \Gamma_i = \Gamma$.
- If $\Gamma_i \vdash^n t : \mathcal{M}_i$ for all $i \in I$ and $I \neq \emptyset$, then $\Gamma \vdash^n t : \mathcal{M}$, where $\mathcal{M} = \sqcup_{i \in I} \mathcal{M}_i$, $n = \sum_{i \in I} n_i$ and $\Gamma = \omega_i \in I \Gamma_i$.

*Proof.* Straightforward by induction on the derivations.

From now on we use the following notation to indicate that we have used the second item.

\[
\left(\Gamma_i \vdash t : \mathcal{M}_i\right)_{i \in I}
\]

\[
\omega_i \in I \Gamma_i \vdash t : \sqcup \mathcal{M}_i
\]

### 4.2. The Characterization of Strong $d\beta$-Normalization

Soundness (Proposition 4.11) is based on Lemma 4.7, based in turn on Lemma 4.3.

**Lemma 4.3** (Substitution lemma). Let $t, u \in T_J$ with $x \in fv(t)$. If both $\Gamma ; x : \mathcal{M} \vdash^n t : \sigma$ and $\Delta \vdash^m u : \mathcal{M}$ hold, then $\Gamma \cup \Delta \vdash^k \{u/x\} t : \sigma$ where $k = n + m - |\mathcal{M}|$.

*Proof.* By induction on the type derivation of $t$. We extend the statement to derivations ending with (MANY), for which the property is straightforward by the $i.h.$

**Case** $t = x$: Then $n = 1$ and by hypothesis $\Gamma = \emptyset$ and $\mathcal{M} = [\sigma]$ (so that $|\mathcal{M}| = 1$). Moreover, $\Delta \vdash^m_{\cap J} u : \mathcal{M}$ necessarily comes from $\Delta \vdash^m_{\cap J} u : \sigma$ by rule (MANY). Let $k = m$, then we conclude $\emptyset \cup \Delta \vdash^{1+m-1} u : \sigma = \Gamma \cup \Delta \vdash^k \{u/x\} x : \sigma$. 
Case \( t = \lambda y.s \) where \( y \neq x \) and \( y \notin \text{fv}(u) \): By definition we have \( \sigma = \mathcal{N} \rightarrow \tau \) and \( \Gamma; x : M; y : \mathcal{N} \vdash \tau \).

By the \( \text{i.h.} \) \( (\Gamma; y : \mathcal{N}) \vdash \Delta \vdash^{k} \{u/x\}s \vdash \tau \) with \( k' = n - 1 + m - |M| \). By the relevance Lemma 4.1 \( y \notin \text{dom}(\Delta) \) so that \( (\Gamma; y : \mathcal{N}) \vdash \Delta = \Gamma \vdash \Delta ; y : \mathcal{N} \). By rule (ABS) we obtain \( \Gamma ; y : \mathcal{N} \vdash \Delta \vdash^{k+1} \lambda y \{u/x\}s : \mathcal{N} \rightarrow \tau \). Let \( k = k' + 1 \). We conclude because \( \lambda y \{u/x\}s = \{u/x\}(\lambda y.s) \) and \( k = k' + 1 = n + m - |M| \).

Case \( t = s(o, y, r) \), where \( y \neq x \) and \( y \notin \text{fv}(u) \): We only detail the case where \( x \in \text{fv}(s) \cap \text{fv}(o) \cap \text{fv}(r) \), the other cases being similar. By definition we have \( \Gamma; x : M_1 \vdash^{n_1} s : \#(\{N_i \rightarrow \tau_i\}_{i \in I}) \), \( \Gamma_2; x : M_2 \vdash^{n_2} o : \#(\{\bigcup_{i \in I} N_i\}) \) and \( \Gamma_3; x : [\tau_i]_{i \in I} \vdash^{n_3} r : \sigma \) where \( \Gamma = \Gamma_1 \uplus \Gamma_2 \uplus \Gamma_3 \), \( M = M_1 \sqcup M_2 \sqcup M_3 \), and \( n = 1 + n_1 + n_2 + n_3 \). Moreover, by Lemma 4.2 we have \( \Delta_1 \vdash^{m_1} u : M_1 \), \( \Delta_2 \vdash^{m_2} u : M_2 \) and \( \Delta_3 \vdash^{m_3} u : M_3 \) where \( \Delta = \Delta_1 \uplus \Delta_2 \uplus \Delta_3 \) and \( m = m_1 + m_2 + m_3 \). The \( \text{i.h.} \) gives \( \Gamma_1 \downarrow \Delta_1 \vdash^{k_1} \{u/x\}s : \#(\{N_i \rightarrow \tau_i\}_{i \in I}) \), \( \Gamma_2 \downarrow \Delta_2 \vdash^{k_2} \{u/x\}o : \#(\{\bigcup_{i \in I} N_i\}) \) and \( \Gamma_3 \downarrow \Delta_3 \vdash^{k_3} \{u/x\}r : \sigma \), where \( k_i = n_i + m_i - |M_i| \) for \( i = 1, 2, 3 \). Then we have a derivation \( \Gamma \downarrow \Delta \vdash \Delta_1 \downarrow \Delta_2 \downarrow \Delta_3 \vdash^{k_1} \{u/x\}s \vdash^{k_2} \{u/x\}o \vdash^{k_3} \{u/x\}r : \sigma \) where \( k = 1 + k_1 + k_2 + k_3 \). We conclude since \( \text{fv}(\{u/x\}) \subseteq \text{fv}(\{u/x\}) \) we have \( \{u/x\}s(o, y, r) = \{u/x\}s\{u/x\}o, y, \{u/x\}r \) and \( k = 1 + k_1 + k_2 + k_3 = 1 + k_1 + k_2 + k_3 = 1 + k_1 + k_2 + k_3 = 1 + k_1 + k_2 + k_3 = 1 + n + m - |M| \).

Lemma 4.4. Let \( t \in T_J \), and \( D \) a list context. Then \( \Gamma \vdash^{n_J} D(\lambda x.t) : \sigma \) if and only if \( \Gamma \vdash^{n_J} \lambda x.D(t) : \sigma \).

\[
\Pi \vdash^{k} s : \#(\{M_i \rightarrow \tau_i\}_{i \in I}) \quad \Delta \vdash^{l} u : \#(\{\bigcup_{i \in I} M_i\}) \quad \Lambda; y : [\tau_i]_{i \in I} \vdash^{m} D'(\lambda x.t) : \sigma
\]

The \( \text{i.h.} \) gives a derivation \( \Lambda; y : [\tau_i]_{i \in I} \vdash^{m} \lambda x.D'(t) : \sigma \) and thus a derivation \( \Lambda; y : [\tau_i]_{i \in I}; x : \mathcal{N} \vdash^{m-1} D'(t) : \rho \). By \( \alpha \)-conversion, \( y \notin \text{fv}(s) \cup \text{fv}(u) \), so that \( y \notin \text{dom}(\Pi \uplus \Delta) \) by Lemma 4.1. We can then build the following derivation of the same size:

\[
\Pi \vdash^{k} s : \#(\{M_i \rightarrow \tau_i\}_{i \in I}) \quad \Delta \vdash^{l} u : \#(\{\bigcup_{i \in I} M_i\}) \quad \Lambda; y : [\tau_i]_{i \in I}; x : \mathcal{N} \vdash^{m} D'(t) : \rho
\]

For the right-to-left implication, we build the first derivations from the second similarly to the previous case.

By nature, subject reduction (or expansion) in the quantitative type system for strong normalization does not hold. Indeed, all subterms are typed, even the ones that will be erased. In most cases, these subterms have free variables, that are typed in the environment. When the term is erased, some bits of the environment are lost which means that the typing is not preserved by reduction steps.

Example 4.5. Let \( t = \lambda x.I(y, z. z) \rightarrow_{d_\beta} I \). The term \( t \) can be typed with the derivation below, with environment \( y : [\sigma] \). However, by relevance, the term \( I \) can only be typed with
an empty environment since that term has no free variables.

\[
\begin{align*}
\vdash x : [\tau] \vdash x : \tau \\
\vdash \lambda x. I : [] \rightarrow [\tau] \rightarrow \tau \\
\vdash y : [\sigma] \vdash y : \sigma \\
\vdash z : [\tau] \rightarrow \tau \vdash z : [\tau] \rightarrow \tau \\
y : [\sigma] \vdash (\lambda x. I)(y, z, z) : [\tau] \rightarrow \tau
\end{align*}
\]

We thus prove subject reduction only for non-erasing steps.

**Definition 4.6** (Erasing step). A reduction step \( t_1 \rightarrow_{d,\beta} t_2 \) is said to be erasing iff the reduced \( d,\beta \)-redex in \( t_1 \) is of the form \( D(\lambda x.t)(u, y, r) \) with \( x \notin \text{fv}(t) \) or \( y \notin \text{fv}(r) \).

**Lemma 4.7** (Non-erasing subject reduction). Let \( \Gamma \vdash_{n_1} t_1 : \sigma \). If \( t_1 \rightarrow_{d,\beta} t_2 \) is a non-erasing step, then \( \Gamma \vdash_{n_2} t_2 : \sigma \) with \( n_1 > n_2 \).

**Proof.** By induction on \( t_1 \rightarrow t_2 \).

**Case** \( t_1 = D_n(\lambda x.t)(u, y, r) \rightarrow_\beta \{D_n(\{u/x\}t)/y\}r = t_2 \): Because the step is non-erasing, the types of \( y \) and \( x \) are not empty by Lemma 4.1, so that we have the following derivation, with \( \Gamma = \psi_{i \in I} \Sigma_i \psi_{i \in I} \Delta_i \psi \Lambda, n_1 = \sum_{i \in I} (n_{i}^1 + n_{i}^2) + n_r + 1 \) and \( I \neq \emptyset \).

\[
\begin{align*}
(\Sigma_i \vdash_{n_i^1} D_n(\lambda x.t) : M_i \rightarrow \tau_i)_{i \in I} & \quad (\Delta_i \vdash_{n_i^2} u : M_i)_{i \in I} \\
\psi_{i \in I} \Sigma_i \psi_{i \in I} \Delta_i \psi \Lambda & \vdash D_n(\lambda x.t)(u, y, r) : \sigma
\end{align*}
\]

For each \( i \in I \), Lemma 4.4 gives a derivation \( \Sigma_i \vdash_{n_i^1} \lambda x.D_n(t) : M_i \rightarrow \tau_i \) and therefore we have a derivation \( \Sigma_i; x : M_i \vdash_{n_i^1} D_n(t) : \tau_i \) where \( n_i^1 = n_i^2 - 1 \). Moreover, the substitution Lemma 4.3 gives \( \Sigma_i \psi_{i \in I} \Delta_i \vdash_{k_i} \{u/x\}D_n(t) : \tau_i \), where \( k_i = n_i^1 + n_i^2 - |M_i| \), so that we have a derivation \( \psi_{i \in I} \Sigma_i \psi_{i \in I} \Delta_i \vdash_{k_i + 1} \{u/x\}D_n(t) : \tau_i \), where \( \psi_{i \in I} \Delta_i \psi \Lambda \vdash M_n(\lambda x.t)(u, y, r) : \sigma \) with \( n_2 = n_r + \sum_{i \in I} k_i < n_1 \).

**Case** \( t_1 = \lambda x.t \rightarrow \lambda x.t' = t_2 \), where \( t \rightarrow t' \): By hypothesis, we have \( \sigma = M \rightarrow \tau \) and \( \psi ; x : M \vdash_{k-1} t' : \tau \) for \( n_1 - 1 > k \). We can build a derivation of size \( n_2 = k + 1 \) and we get \( n_1 > n_2 \).

**Case** \( t_1 = t(u, x, r) \) and the reduction is internal: By hypothesis, we have the derivations \( \Sigma \vdash_{n_1} t : \#([M_i \rightarrow \tau_i]_{i \in I}), \Delta \vdash_{n_2} u : \#([\lambda_{i \in I} M_i]) \) and \( \psi ; x : [\tau_i]_{i \in I} \vdash_{r} r : \sigma \) with \( \Sigma = \psi \Delta \psi \Lambda \) and \( n_1 = 1 + n_t + n_u + n_r \).

**Subcase** \( t_1 \rightarrow t'(u, x, r) = t_2 \), where \( t \rightarrow t' \): If \( I \neq \emptyset \), we have \( \Sigma = \psi_{i \in I} \Sigma_i, n_t = \sum_{i \in I} n_i^i \) and derivations \( \Sigma_i \vdash_{n_i^1} t : M_i \rightarrow \tau_i \). If \( I = \emptyset \), we have \( \#(M_i \rightarrow \tau_i)_{i \in I} = [\tau] \) and a derivation \( \Sigma \vdash_{n_i^1} t : \tau \). In both cases, we apply the i.h. and derive \( \Sigma \vdash_{k'} t' : \#([M_i \rightarrow \tau_i]_{i \in I}) \) with \( k < n_t \). We can build a derivation of size \( n_2 = 1 + k + n_u + n_r \) and we get \( n_1 > n_2 \).

**Subcase** \( t_1 \rightarrow t(u', x, r) = t_2 \), where \( u \rightarrow u' \): Let \( \#([\lambda_{i \in I} M_i] = [\rho_j]_{j \in J} \). In particular, if \( \square \in I, M_i = [\square] \), then \( J \) is a singleton. We have \( \Delta = \psi_{j \in J} \Delta_j, n_u = \sum_{j \in J} n_j^2 \) and derivations \( \Delta_j \vdash_{n_j} \rho_j \). We apply the i.h. and derive \( \Delta \vdash_{k'} u : \#([\lambda_{i \in I} M_i]) \) with \( k < n_u \). We can build a derivation of size \( n_2 = 1 + n_t + n_u + k \) and we get \( n_1 > n_2 \).

**Subcase** \( t_1 \rightarrow t(u, x, r') = t_2 \), where \( r \rightarrow r' \): By the i.h. we have \( \Delta ; x : [\tau_i]_{i \in I} \vdash_{k'} r : \sigma \) with \( k < n_r \). We can build a derivation of size \( n_2 = 1 + n_t + n_u + k \) and we get \( n_1 > n_2 \).
Although subject reduction does not always hold, the characterization of normalizable terms as typable should. To prove this, we need a weaker form of subject reduction: the fact that the right-hand term of an erasing reduction is still typed. This is the goal of the following lemma. Notice that we do not consider any reductio non contrarii, but one occurring inside a left-right \( R \). We will use the syntax of terms given in Equation 3.1 to conclude the proof (Lemma 4.10).

**Lemma 4.8.** Let \( t = D_n(λx.s)(u, y,r) \) and \( t' = \{ D_n(\{u/x\}s)/y\}r \) such that \( Γ \vdash^{k}\Gamma \cap J W(t) : σ \). Then,

1. If \( y \notin \text{fv}(r) \), then there are typing derivations for \( W(t') = W(r), D_n(s) \) and \( u \) having measures \( k_{W(t')} \), \( k_{D_n(s)} \) and \( k_u \) resp. such that \( k > 1 + k_{W(t')} + k_{D_n(s)} + k_u \).
2. If \( y \in \text{fv}(r) \) and \( x \notin \text{fv}(s) \), then there are typing derivations for \( W(t') = W(\{D_n(s)/y\})r \) and \( u \) having measures \( k_{W(t')} \) and \( k_u \) resp. such that \( k > 1 + k_{W(t')} + k_u \).

**Proof.** We prove a stronger statement: the derivation for \( W(t') \) is of the shape \( Γ' \vdash^{k'}W(t') : σ \) with the same \( σ \) but \( Γ' \subseteq Γ \). We proceed by induction on \( R \):

**Case** \( R = \emptyset \):
1. The derivation of \( t \) has premises \( Γ_λ \vdash^{k_λ} D_n(λx.s) : τ, Δ \vdash^{k_u} u : ρ \) and \( λ \vdash^{k_τ} r : σ \), for some appropriate \( τ \) and \( ρ \), such that \( Γ = Γ_λ \cup Δ \cup λ \) and \( k = k_λ + k_u + k_v + 1 \). By Lemma 4.4, we have a derivation \( Γ_λ \vdash^{k_λ} λx.D_n(s) : τ \).
2. Then, \( τ = M \to τ' \) with \( M \) potentially empty and we have a derivation \( Γ_λ : x : M \vdash^{k_λ-1} D_n(s) : τ' \). Let \( k_{D_n(s)} = k_λ - 1 \). We have \( k > k + 1 + k_v + k_{D_n(s)} + k_u \) and we let \( Γ' = λ \) since \( t' = τ \). We can conclude since \( Γ' \subseteq Γ \).

**Case** \( R = R'(u', \bar{z}, r') \):
1. The derivation of \( R(t) \) has premises \( Γ_1 \vdash^{k_{R'(t)}} \) having measures \( k_{D_n(s)} \) and \( k_u \).
2. By i.h. we get from the first premise:
3. In case (1) a typing derivation for \( D_n(s) \) of measure \( k_{D_n(s)} \) and the fact that \( k_{R'(t)} > 1 + k_{R'(t)} + k_{D_n(s)} + k_u \).
4. In case (2) \( 1 + k_{R'(t)} + k_u \).

Using the type derivations for \( R'(t'), u' \) and \( r' \) we can build a derivation \( Γ_2 \vdash^{k_{R'(t')}} R'(t')(u', \bar{z}, r') : σ \), where \( k_{R'(t')} = 1 + k_{R'(t')} + k_{D_n(s)} + k_u \) and \( k_u + k_r + 1 = 1 + k_{R'(t)} + k_{D_n(s)} + k_u \). In case (1) we can conclude because \( k = 1 + k_{R(t)} + k_{u'} + k_{r'} > l.h. 1 + (1 + k_{R'(t)} + k_{D_n(s)} + k_u) + k_{u'} + k_r = 1 + k_{R(t)} + k_{D_n(s)} + k_u \).
Case $R = n(u', z.R')$: The derivation of $R(t)$ has premises: $\Gamma_n \vdash_{k_n} n : \#([M_i \rightarrow \tau_i]_{i \in I})$, $\Delta \vdash_{k_u} u' : \#(\bigcup_{i \in I} M_i)$ and $\Lambda_1; z : [\tau_i]_{i \in I} \vdash_{k^{(v)}(t)} R'(t) : \sigma$. We have $\Delta = \Delta_n \uplus \Delta \uplus \Lambda_1$ and $k = 1 + k_n + k_u + k^{(r)}(t)$. By the i.h., we get from the third premise:

1. In cases (1) and (2) a derivation $\Lambda_2; z : [\tau_i]_{i \in I} \vdash_{k^{(v)}(t)} R'(t) : \sigma$ such that $\Delta_2 \subseteq \Delta_1$, and $I' \subseteq I$ ($I'$ possibly empty), and a typing derivation for $u$ of measure $k_u$.

2. In case (1) a typing derivation for $D_n(s)$ of measure $k_{D_n(s)}$ and the fact that $k^{(r)}(t) > 1 + k_{R'(t')} + k_{D_n(s)} + k_u$.

3. In case (2) $1 + k_{R'(t')} + k_u$.

To build a derivation for $R(t')$, we need in particular derivations of type $\#([M_i \rightarrow \tau_i]_{i \in I'})$ for $n$ and $\#(\bigcup_{i \in I'} M_i)$ for $u'$.

Subcase $I' \neq \emptyset$: Then $\#([M_i \rightarrow \tau_i]_{i \in I'}) = \#([M_i \rightarrow \tau_i]_{i \in I'})$ and by Lemma 4.2 it is possible to construct a derivation $\Gamma_n' \vdash_{k'} n : [M_i \rightarrow \tau_i]_{i \in I'}$ from the original one for $u'$ we build a derivation $\Delta' \vdash_{k'} u' : \#(\bigcup_{i \in I'} M_i)$ verifying $\Delta' \subseteq \Delta$ and $k'_u \leq k_u$. There are three cases:

Subcase: $(M_i)_{i \in I}$ are all empty, and therefore $(M_i)_{i \in I'}$ are all empty. Then we set $\#(\bigcup_{i \in I'} M_i) = \#(\bigcup_{i \in I'} M_i)$. We take the original derivation so that $\Delta' = \Delta$, $k'_u = k_u$.

Subcase: $(M_i)_{i \in I'}$ are all empty but $(M_i)_{i \in I}$ are not all empty. As a consequence, $\bigcup_{i \in I} M_i \neq \emptyset$ and we take an arbitrary type $\rho$ of $\bigcup_{i \in I} M_i$, as a witness for $u'$, so that, $\Delta_\rho \vdash_{k_R} u' : \rho$ holds by Lemma 4.2. We have the expected derivation with rule $\langle \text{ANY} \rangle$ taking $\Delta' = \Delta_\rho$, $\#(\bigcup_{i \in I'} M_i) = [\rho]$ and $k'_u = k_\rho$.

Subcase: $\#(\bigcup_{i \in I'} M_i) = \#(\bigcup_{i \in I'} M_i)$. By Lemma 4.2 it is possible to construct the expected derivation from the original ones for $u'$.

Finally, we conclude by the following derivation for $R(t')$:

$$
\begin{align*}
\Gamma_n' \vdash_{k'} n : [M_i \rightarrow \tau_i]_{i \in I'} & \quad \Delta' \vdash_{k'} u' : \#(\bigcup_{i \in I'} M_i) \\
\end{align*}
$$

where $\Phi = \Lambda_2; z : [\tau_i]_{i \in I'} \vdash_{k^{(v)}(t')} R'(t') : \sigma$, $\Delta' = \Delta_n' \uplus \Delta' \uplus \Lambda_2$, and the total measure of the derivation is $k_{R^{(t')}} = 1 + k_n' + k_u' + k^{(r)}(t')$. We have $k > 1 + k_n' + k_u' + k^{(r)}(t') > i.h.$, $1 + k_n' + k_u' + 1 + k_{R^{(t')}} + k_{D_n(s)} + k_u > 1 + k_{R^{(t')}} + k_{D_n(s)} + k_u$ in case (1). Similarly but without $k_{D_n(s)}$ in case (2). We can conclude since $\Gamma' \subseteq \Gamma$.

Case $I = I' = \emptyset$: We are done by taking the original derivations.

Case $I \neq \emptyset = I'$: Let us take an arbitrary $j \in I$: the type $[M_j \rightarrow \tau_j]$ is set as a witness for $n$, whose derivation $\Gamma_n \vdash_{k_n} n' : [M_j \rightarrow \tau_j]_{j \in I}$ is obtained from the derivation $\Gamma_n \vdash_{k_n} n : \#([M_i \rightarrow \tau_i]_{i \in I})$ by the split Lemma 4.2. For $u'$, we take as a witness an arbitrary $\rho \in \#(\bigcup_{i \in I} M_i)$ and we set $\#(\bigcup_{i \in I} M_i) = [\rho]$. If $\bigcup_{i \in I} M_i = []$, then $\rho$ is the original witness. Otherwise $\rho$ is a type of one of the $M_i$'s. In both cases we use the split Lemma 4.2 to get a derivation $\Delta' \vdash_{k'_u} u' : [\rho]$ where $\Delta' \subseteq \Delta$ and $k'_u \leq k_u$. Using the type derivation given by the i.h. for $R'(t')$, we conclude by the following derivation for $R(t')$:

$$
\begin{align*}
\Gamma_n' \vdash_{k_n'} n' : [M_j \rightarrow \tau_j] & \quad \Delta' \vdash_{k'_u} u' : [\rho] \\
\end{align*}
$$

where $\Gamma_n' \subseteq \Gamma_n$, $\Delta' \subseteq \Delta$, $k'_n \leq k_n$, $k'_u \leq k_u$. We have $\Gamma' = \Gamma_n' \uplus \Delta' \uplus \Lambda_2 \subseteq \Gamma$. 

In case (1) we can conclude because \( k = 1 + k_n + k_n' + k_n'(t) > 1 + k_n' + k_n' + (1 + k_n'(t') + k_{D_n(s)} + k_u) = 1 + k_n(t') + k_{D_n(s)} + k_u \). Similarly but without \( k_{D_n(s)} \) in case (2).

We now finish the proof of soundness by proving that all typable terms have a finite reduction length, that is bounded by the maximum number of \( d \)-steps until normal form. This maximal length is written \( ||t||_{d,\beta} \) for a term \( t \).

**Lemma 4.9.** The following equalities hold:

\[
\begin{align*}
||x||_{d,\beta} & = 0 \\
||\lambda x. t||_{d,\beta} & = ||t||_{d,\beta} \\
||n(u, x.r)||_{d,\beta} & = ||n||_{d,\beta} + ||u||_{d,\beta} + ||r||_{d,\beta} \\
||R(D_n(\lambda x.s)(u, y.r))||_{d,\beta} & = \begin{cases} 
1 + ||R||_{d,\beta} + ||D_n(s)||_{d,\beta} + ||u||_{d,\beta} & \text{if } y \notin \text{fv}(r); \\
1 + ||R||_{d,\beta} + ||u||_{d,\beta} & \text{if } x \notin \text{fv}(s); \\
1 + ||R||_{d,\beta} + ||u||_{d,\beta} & \text{if } x \in \text{fv}(s)
\end{cases}
\end{align*}
\]

**Proof.** A consequence of Definition 3.7 and Theorem 3.11.

**Lemma 4.10.** If \( \Gamma \models^k_{\gamma, l} t : \sigma \), then \( ||t||_{d,\beta} \leq k \).

**Proof.** We proceed by induction on \( k \) and we reason by case analysis on \( t \) according to the alternative grammar (Equation 3.1 on Page 12).

**Case** \( t = x \): The derivation is just an axiom and \( k = 1 \), so that \( ||x||_{d,\beta} = 0 < 1 = k \).

**Case** \( t = \lambda x. u \): There is a typing derivation for \( u \) of size \( k - 1 < k \). The \( i.h. \) gives \( ||u||_{d,\beta} \leq k - 1 \), so that \( ||t||_{d,\beta} = ||u||_{d,\beta} \leq k \).

**Case** \( t = n(u, x.r) \): There are typings of \( n, u \) and \( r \) with measures \( k_n, k_u \) and \( k_r \), resp. such that \( k = 1 + k_n + k_u + k_r \). We then get \( ||t||_{d,\beta} = ||n||_{d,\beta} + ||u||_{d,\beta} + ||r||_{d,\beta} \leq i.h. k_n + k_u + k_r \leq k \).

**Case** \( t = R(D_n(\lambda x.s)(u, y.r)) \): There are three possible cases:

**Subcase** \( x \in \text{fv}(s) \) and \( y \in \text{fv}(r) \): Then \( t \rightarrow_{d,\beta} R\{D_n(\{u/x\}s)/y\}r \) = \( t_0 \) and \( ||t||_{d,\beta} = 1 + ||t_0||_{d,\beta} \). Moreover, the subject reduction Lemma 4.7 gives \( \Gamma \models^k_{\gamma, l} t_0 : \sigma \) with \( k' < k \). By the \( i.h. \) we have \( ||t_0||_{d,\beta} \leq k' \). Thus we conclude \( ||t||_{d,\beta} = 1 + ||t_0||_{d,\beta} \leq 1 + k' \leq k \).

**Subcase** \( y \notin \text{fv}(r) \): Then \( t \rightarrow_{d,\beta} R\{\{u/x\}s/y\}r \) = \( t_0 \) and \( ||t||_{d,\beta} = 1 + ||t_0||_{d,\beta} + ||D_n(s)||_{d,\beta} + ||u||_{d,\beta} \). By subject reduction for erasing steps (Lemma 4.8) there are typings of \( t_0, D_n(s) \) and \( u \) having measures \( k_{t_0}, k_{D_n(s)} \) and \( k_u \) resp. such that \( k > 1 + k_{t_0} + k_{D_n(s)} + k_u \). Thus we conclude \( ||t||_{d,\beta} = 1 + ||t_0||_{d,\beta} + ||D_n(s)||_{d,\beta} + ||u||_{d,\beta} \leq i.h. 2 + k_{t_0} + k_{D_n(s)} + k_u \leq k \).

**Subcase** \( x \notin \text{fv}(s) \) and \( y \notin \text{fv}(r) \): Then \( t \rightarrow_{d,\beta} R\{D_n(\{s/y\}r) \) = \( t_0 \) and \( ||t||_{d,\beta} = 1 + ||t_0||_{d,\beta} + ||u||_{d,\beta} \). By subject reduction for erasing steps (Lemma 4.8) there are typings of \( t_0 \) and \( u \) having measures \( k_{t_0} \) and \( k_u \) resp. such that \( k > 1 + k_{t_0} + k_u \). Thus we conclude \( ||t||_{d,\beta} = 1 + ||t_0||_{d,\beta} + ||u||_{d,\beta} \leq i.h. 1 + k_{t_0} + k_u < k \).

As a corollary we obtain:

**Proposition 4.11** (Soundness for \( \lambda J_n \)). If \( t \in \text{\ihr J-typable} \), then \( t \in \text{SN}(d,\beta) \).

The completeness Lemma 4.15 is based on typability of normal forms (Lemma 4.12) and non-erasing subject expansion (Lemma 4.14). This last one is based itself on anti-substitution (Lemma 4.13).
Lemma 4.12 (Typing normal forms).

(1) For all \( t \in \text{NF}_{d,3} \), there exists \( \Gamma, \sigma \) such that \( \Gamma \vdash \tau ; t : \sigma \).

(2) For all \( t \in \text{NE}_{d,3} \), for all \( \sigma \), there exists \( \Gamma \) such that \( \Gamma \vdash \tau ; t : \sigma \).

Proof. By simultaneous induction on \( t \in \text{NF}_{d,3} \) and \( t \in \text{NE}_{d,3} \).

First, the cases relative to statement (1).

Case \( t = x \): Pick an arbitrary \( \sigma \). We have \( x : [\sigma] \vdash x : \sigma \) by rule (VAR).

Case \( t = \lambda x.s \) where \( s \in \text{NF}_{d,3} \): By i.h. on \( s \) there exists \( \Gamma' \) and \( \tau \) such that \( \Gamma' \vdash \tau ; s : \tau \). Let \( \Gamma \) and \( \mathcal{N} \) be such that \( \Gamma' = \Gamma ; x : \mathcal{N} \) (\( \mathcal{N} \) is possibly empty). We get \( \Gamma \vdash \lambda x.s : \mathcal{N} \rightarrow \tau \) by rule (ABS). We conclude by taking \( \sigma = \mathcal{N} \rightarrow \tau \).

Case \( t = s(u, y, r) \) where \( u, r \in \text{NF}_{d,3} \) and \( \mathcal{N} \in \text{NE}_{d,3} \): By the i.h. on \( r \) there is a derivation of \( \mathcal{N}' \vdash \tau ; \sigma \). Let \( \Lambda \) and \([\tau_i]_{i \in I}\) be such that \( \mathcal{N}' = \Lambda ; y : [\tau_i]_{i \in I} \). Now we construct a derivation \( \Pi \vdash s : \#([\tau_i]_{i \in I}) \) as follows:

- If \( I = \emptyset \), then the i.h. on \( s \) gives a derivation \( \Pi \vdash s : \tau \) and we use rule (many) to get \( \Pi \vdash s : \mathcal{N} \). We conclude by setting \( \#([\tau_i]_{i \in I}) = \mathcal{N} \).
- If \( I \neq \emptyset \), then the i.h. on \( s \) gives a derivation of \( \Pi_i \vdash s : [\tau_i]_{i \in I} \). We take \( \Pi = \cup_{i \in I} \Pi_i \) and we conclude with rule (many) since \( \#([\tau_i]_{i \in I}) = \mathcal{N} \).

Finally, the i.h. on \( u \) gives a derivation \( \Delta \vdash u : \rho \) from which we get \( \Delta \vdash u : [\rho] \), by choosing \( \#([\tau_i]_{i \in I}) = [\rho] \). We conclude with rule (app) as follows:

\[
\Pi \vdash s : \#([\tau_i]_{i \in I}) \quad \Delta \vdash u : \#([\tau_i]_{i \in I}) \quad \Lambda ; y : [\tau_i]_{i \in I} \vdash \tau ; \sigma \\
\Pi \cup \Delta \vdash \Lambda \vdash \Pi \vdash s(u, y, r) : \sigma
\]

Next, the cases relative to statement (2).

Case \( t = x \): As seen above, given an arbitrary type \( \sigma \), we can take \( \Gamma = [\sigma] \).

Case \( t = s(u, y, r) \) where \( u \in \text{NF}_{d,3} \) and \( \mathcal{N} \in \text{NE}_{d,3} \): Pick an arbitrary \( \sigma \). The proof proceeds \( \text{ipsis verbis} \) as in the case \( t = s(u, y, r) \) above.

Lemma 4.13 (Anti-substitution). If \( \Gamma \vdash \{ u/x \} : \sigma \) where \( x \in \text{fv}(t) \), then there exist \( \Gamma, \Gamma_u \) and \( \mathcal{M} \neq [] \) such that \( \Gamma ; x : \mathcal{M} \vdash t : \sigma \), \( \Gamma_u \vdash u : \mathcal{M} \) and \( \Gamma \vdash \Gamma \cup \Gamma_u \).

Proof. By induction on the derivation \( \Gamma \vdash \{ u/x \} : \sigma \). We extend the statement to derivations ending with (many), for which the property is straightforward by the i.h. We reason by cases on \( t \).

Case \( t = x \): Then \( \{ u/x \} t = u \). We take \( \Gamma_t = \emptyset, \Gamma_u = \Gamma, \mathcal{M} = [\sigma] \), and we have \( x : [\sigma] \vdash x : \sigma \) by rule (var) and \( \Gamma \vdash u : \mathcal{M} \) by rule (many) on the derivation of the hypothesis.

Case \( t = \lambda y.s \) where \( y \neq x \) and \( y \notin \text{fv}(u) \) and \( x \in \text{fv}(s) \): Then \( \{ u/x \} t = \lambda y.\{ u/x \} s \). We have \( \sigma = \mathcal{N} \rightarrow \tau \) and \( \Gamma ; y : \mathcal{N} \vdash \{ u/x \} s : \tau \).

By the i.h. there exists \( \Gamma' \), \( \Gamma_u \), \( \mathcal{M} \neq [] \) such that \( \Gamma' ; y : \mathcal{N} ; x : \mathcal{M} \vdash s : \tau, \Gamma_u \vdash u : \mathcal{M} \), and \( \Gamma ; y : \mathcal{N} = (\Gamma' ; y : \mathcal{N} ; y : \mathcal{M} \vdash s : \tau) \). Moreover, by \( \alpha \)-conversion and Lemma 4.1 we know that \( y \notin \text{dom}(\Gamma_u) \) so that \( \Gamma = \Gamma' \cup \Gamma_u \). We conclude by deriving \( \Gamma' ; y : \mathcal{N} \vdash \lambda x : \mathcal{M} \rightarrow \tau \) with rule (ABS). Indeed, by letting \( \Gamma_t = \Gamma' \) we have \( \Gamma = \Gamma_t \cup \Gamma_u \) as required.

Case \( t = t_1(t_2, y, r) \), where \( y \neq x \), \( y \notin \text{fv}(u) \) and \( x \in \text{fv}(t_1) \cup \text{fv}(t_2) \cup \text{fv}(r) \) \( y \): We detail the case where \( x \in \text{fv}(t_1) \cup \text{fv}(t_2) \cup \text{fv}(r) \), the other cases are similar. By construction, we have derivations \( \Gamma_1 \vdash \{ u/x \} t_1 : \#([\mathcal{N}_1 \rightarrow \tau_i]_{i \in I}) \), \( \Gamma_2 \vdash \{ u/x \} t_2 : \#([\cup_{i \in I} \mathcal{N}_1]) \) and \( \Gamma_3 ; y : [\tau_i]_{i \in I} \vdash \{ u/x \} r ; \sigma \), with \( \Gamma = \Gamma_1 \cup \Gamma_2 \cup \Gamma_3 \).

By the i.h. there are environments \( \Gamma_{t_1}, \Gamma_{t_2}, \Gamma, \Gamma_u, \Gamma_u^2, \Gamma_u^3 \) and multitypes \( \mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3 \) all different from \( [] \) such that \( \Gamma_t ; x : \mathcal{M}_1 \vdash t_1 : \#([\mathcal{N}_1 \rightarrow \tau_i]_{i \in I}) \), \( \Gamma_{t_2} ; x : \mathcal{M}_2 \vdash t_2 : \#([\cup_{i \in I} \mathcal{N}_1]) \), \( \Gamma_r ; x : \mathcal{M}_3 \vdash r ; \sigma \), \( \Gamma_u^1 \vdash u : \mathcal{M}_1 \), \( \Gamma_u^2 \vdash u : \mathcal{M}_2 \), \( \Gamma_u^3 \vdash u : \mathcal{M}_3 \) and
\[
\Gamma_1 = \Gamma_t \uplus \Gamma_1^0, \quad \Gamma_2 = \Gamma_t \uplus \Gamma_2^0, \quad \Gamma_3 = \Gamma_r \uplus \Gamma_3^0. \]

Let \( \Gamma_t = \Gamma_{t_1} \uplus \Gamma_{t_2} \uplus \Gamma_r, \quad \Gamma_u = \Gamma_u^1 \uplus \Gamma_u^2 \uplus \Gamma_u^3 \)
and \( M = M_1 \uplus M_2 \uplus M_3. \) We can build a derivation \( \Gamma_t; x: M \vdash t_{1(y,r)}: \sigma \)
with rule (APP) and a derivation \( \Gamma_u \vdash u: M \) with Lemma 4.2. We conclude since
\( \Gamma = \Gamma_1 \uplus \Gamma_2 \uplus \Gamma_3 = \Gamma_{t_1} \uplus \Gamma_{t_2} \uplus \Gamma_r \uplus \Gamma_u \uplus \Gamma_u^3 = \Gamma_t \uplus \Gamma_u. \)

**Lemma 4.14** (Non-erasing subject expansion). If \( \Gamma \models r_J t_2 : \sigma \) and \( t_1 \rightarrow_{d\beta} t_2 \) is a non-erasing step, then \( \Gamma \models r_J t_1 : \sigma. \)

**Proof.** By induction on \( t_1 \rightarrow_{d\beta} t_2. \)

**Case** \( t_1 = \text{D}(\lambda x.t)(u,y.r) \rightarrow_{\beta} \{ \text{D}(\langle u/x \rangle t)/y \} r = t_2: \) Since the reduction is non-erasing, we have \( y \in \text{fv}(r) \) and \( x \in \text{fv}(t). \) By Lemma 4.13, there exists \( \Gamma_r, \Gamma' \) and \( \mathcal{N} \) such that \( \Gamma_r; y : \mathcal{N}; r : \sigma, \quad \Gamma' \models \{ \langle u/x \rangle t \} : \mathcal{N} \) and \( \Gamma = \Gamma' \uplus \Gamma_r. \) Let \( \mathcal{N} = \{ \tau_i \}_{i \in I} \neq [\ ] \) since \( y \in \text{fv}(r). \) By rule (\text{Many}), we have a decomposition \( (\Gamma'_i) \models \{ \langle u/x \rangle t \} : \tau_i \) with \( \Gamma'_r = \emptyset \). By rule (\text{Abs}) followed by (\text{Many}), there are derivations \( \Gamma_I \models \lambda x.\text{D}(t) : [M_i \rightarrow \tau_i]_{i \in I} \)
with \( \Gamma_I = \{ \psi_i \}_{i \in \mathcal{I}^I}. \) By Lemma 4.4, there is a derivation \( \Gamma_I \models \text{D}(\lambda x.t) : [M_i \rightarrow \tau_i]_{i \in I}. \)
Finally, by Lemma 4.2, there is a derivation \( \Gamma_u \vdash u : \psi_i \in \mathcal{M}_i \) with \( \Gamma_u = \{ \psi_i \}_{i \in \mathcal{I}^I}. \)
Since neither \( I \) nor the \( \mathcal{M}_i \)'s are empty, the choice operator in both cases the identity and we can build the following derivation using rule (\text{App}):

\[
\Gamma_i \models \text{D}(\lambda x.t) : [M_i \rightarrow \tau_i]_{i \in I} \quad \Gamma_u \vdash u : \psi_i \in \mathcal{M}_i \quad \Gamma_r \vdash y : [\tau_i]_{i \in I} : r : \sigma
\]

\[
\Delta \models \text{D}(\lambda x.t)(u,y,r) : \sigma
\]

We verify \( \Gamma = \Gamma' \uplus \Gamma_r = \emptyset \uplus \{ \psi_i \}_{i \in \mathcal{I}^I} \) and \( \Gamma_r = \emptyset \uplus \{ \psi_i \}_{i \in \mathcal{I}^I} \downarrow \Gamma_r = \Gamma_t \uplus \Gamma_u. \)

**Case** \( t_1 = \lambda x.t \) and \( t_1 = t(u,x,r) \) and the reduction is internal: These cases are direct by the \( i.h. \)

We cannot conclude completeness straightforward, given that subject expansion was only
shown for non-erasing cases. Instead, we prove that from any term on the right of a
reduction, we can build a derivation for the term on the left. We rely on the previous lemma
for the non-erasing steps, and construct derivations for erasing cases, in which the typing
environment grows with anti-reduction. We use the inductive characterization of strong
normalization \( ISN(d\beta) \) to recognize the left terms that are indeed strongly normalizing,
which are the only ones for which we can build a typing derivation.

**Lemma 4.15** (Completeness for \( \lambda J_n \)). If \( t \in \text{SN}(d\beta) \), then \( t \) is \( J \)-typable.

**Proof.** In the statement, we replace \( \text{SN}(d\beta) \) by \( ISN(d\beta), \) using Theorem 3.11. We use induction
on \( ISN(d\beta) \) to show the following stronger property \( \mathcal{P}: \) If \( t \in ISN(d\beta) \) then there
are \( \Gamma, \sigma \) such that \( \Delta \models t : \sigma, \) and if \( t \in n, \) then the property holds for any \( \sigma. \)

**Case** \( t = x: \) We get \( \sigma = \{ x \} : x : \sigma \).

**Case** \( t = \lambda x.s, \) where \( s \in ISN(d\beta) \): By the \( i.h. \) we have \( \Delta \vdash s : \tau. \) Let us write \( \Delta \) as
\( \Gamma; x : \mathcal{M}, \) where \( \mathcal{M} \) is possibly empty. Then we get \( \Gamma \models \lambda x.s : \sigma, \) where \( \sigma = \mathcal{M} \rightarrow \tau, \)
by using rule (\text{Abs}) (the previous derivation).

**Case** \( t = n(u,x,r), \) where \( n, u, r \in ISN(d\beta) \) and \( r \in \text{NF}_n \): By the \( i.h. \) there are derivations
\( \Delta \vdash u : \rho \) and \( \Lambda; x : [\tau_i]_{i \in I} \vdash r : \sigma \) with \( I \) possibly empty. Moreover, \( \Delta \vdash u : [\rho] \) holds
by rule (\text{Many}). If \( r \in n, \) we have a derivation for any type \( \sigma \) by the strong \( i.h. \)

We now construct a derivation \( \Pi \vdash n : \#([\tau]_{i \in I}) \) as follows:

- If \( I = \emptyset, \) then the \( i.h. \) gives \( \Pi \vdash n : \tau \) for an arbitrary \( \tau, \) and then we obtain
  \( \Pi \vdash n : [\tau] \) by rule (\text{Many}). We conclude by setting \( \#([\tau]_{i \in I}) = [\tau]. \)
If \( I \neq \emptyset \), then by the stronger \( i.h. \) we can derive \( \Pi_i \vdash n : [|] \rightarrow \tau_i \) for each \( i \in I \). We take \( \Pi = \cup_{i \in I} \Pi_i \) and we conclude with rule (\textsc{many}) since \( \#([|] \rightarrow \tau_i)_{i \in I} = [|] \rightarrow \tau_{i_{\epsilon}} \).

We conclude with rule (\textsc{app}) as follows, by setting in particular \( \#([\cup_{i \in I}])] = [\rho] \).

\[
\begin{array}{cccc}
\Pi \vdash n : \#([|] \rightarrow \tau_{i_{\epsilon}}) & \Delta \vdash u : \#([\cup_{i \in I}])] & \Lambda ; y : [\tau_{i_{\epsilon}}]_{i \in I} \vdash r : \sigma \\
\Pi \cup \Delta \cup \Lambda \vdash s(u, y.r) : \sigma
\end{array}
\]

**Case** \( t \notin \text{NF}_0 \): That is, \( t = R(\mathcal{D}_n(\lambda x.s)(u, y.r)) \), where \( t' = R(\{u/x\} \mathcal{D}_n(s)/y)r \in \text{ISN}(d\beta) \), \( \mathcal{D}_n(s) \in \text{ISN}(d\beta) \), and \( u \in \text{ISN}(d\beta) \). Notice that \( t \notin \emptyset \) by Lemma 4.1. By the \( i.h. \) \( t', \mathcal{D}_n(s) \) and \( u \) are typable. We show by a second induction on \( R \) that \( \Sigma \vdash t' : \sigma \) implies \( \Gamma \vdash t : \sigma \), for some \( \Gamma \). For the base case \( R = \circ \), there are three cases.

**Subcase** \( x \notin \text{fv}(s) \) and \( y \notin \text{fv}(r) \): Since \( t' = \{u/x\} \mathcal{D}_n(s)/y)r \) is typable and \( t \rightarrow_{\beta} t' \), then \( t \) is also typable with \( \Sigma \) and \( \sigma \) by the non-erasing subject expansion Lemma 4.14. We conclude with \( \Gamma = \Sigma \).

**Subcase** \( x \in \text{fv}(s) \) and \( y \in \text{fv}(r) \): Then \( t' = \{u/x\} \mathcal{D}_n(s)/y)r \) is typable and \( t \rightarrow_{\beta} t' \). Since \( x \notin \text{ISN}(d\beta) \), by Lemma 4.1, we have \( \Pi \neq \emptyset \) by Lemma 4.1 since \( y \in \text{fv}(r) \). By the Split Lemma 4.2 there are derivations \( \Pi_i \vdash \mathcal{D}_n(s) : \sigma_i \) such that \( \Pi = \cup_{i \in I} \Pi_i \). Let \( \Lambda = \{\sigma_i\}_{i \in I} \). Then by Lemma 4.4 applied for each \( i \in I \), we retrieve \( \Pi_i \vdash \mathcal{D}_n(s) : [|] \rightarrow \sigma_{i_{\epsilon}} \).

Finally, since \( \#([|] \rightarrow \sigma_{i_{\epsilon}})_{i \in I} = [|] \rightarrow \sigma_{i_{\epsilon}} \), it is sufficient to set \( \#([\cup_{i \in I}])] = [\rho] \) and we obtain the following derivation:

\[
\begin{array}{cccc}
\Pi \vdash \mathcal{D}_n(\lambda x.s) : \#([|] \rightarrow \sigma_{i_{\epsilon}})_{i \in I} & \Delta \vdash u : \#([\cup_{i \in I}])] & \Lambda ; y : [|] \vdash r : \sigma \\
\Gamma \vdash \mathcal{D}_n(\lambda x.s)(u, y.r) : \sigma
\end{array}
\]

where \( \Gamma = \Pi \cup \Delta \cup \Lambda \). We then conclude.

**Subcase** \( y \notin \text{fv}(r) \): Since \( t' = \{u/x\} \mathcal{D}_n(s)/y)r \) is typable and \( t' = r \), then there is a derivation \( \Delta \vdash u : \rho \) where \( y \notin \text{dom}(\lambda) \) holds by relevance (so that \( \Sigma = \Lambda \)). We can then write \( \Lambda ; y : [|] \vdash r : \sigma \). We construct a derivation of \( t \) ending with rule (\textsc{app}). For this we need two witness derivations for \( u \) and \( \mathcal{D}_n(\lambda x.s) \). Since \( u \in \text{ISN}(d\beta) \), the \( i.h. \) gives a derivation \( \Delta \vdash u : \rho \), and then we get \( \Delta \vdash u : [\rho] \) by application of rule (\textsc{many}). Similarly, since \( \mathcal{D}_n(s) \in \text{ISN}(d\beta) \), the \( i.h. \) gives a derivation \( \Pi ; x : \mathcal{M} \vdash \mathcal{D}_n(s) : \tau \) where \( \mathcal{M} \) can be empty. Thus \( \Pi \vdash \lambda x.\mathcal{D}_n(s) : \mathcal{M} \rightarrow \tau \). By Lemma 4.4, we get \( \Pi \vdash \mathcal{D}_n(\lambda x.s) : \mathcal{M} \rightarrow \tau \), and then we get \( \Pi \vdash \mathcal{D}_n(\lambda x.s) : [\mathcal{M} \rightarrow \tau] \) by application of rule (\textsc{many}). Finally, by setting \( \#([|]) = [\mathcal{M} \rightarrow \tau] \) and \( \#([\cup[|]]) = [\rho] \) we construct the following derivation:

\[
\begin{array}{cccc}
\Pi \vdash \mathcal{D}_n(\lambda x.s) : \#([|]) & \Delta \vdash u : \#([\cup[|]]) & \Lambda ; y : [|] \vdash r : \sigma \\
\Gamma \vdash \mathcal{D}_n(\lambda x.s)(u, y.r) : \sigma
\end{array}
\]

where \( \Gamma = \Pi \cup \Delta \cup \Lambda \). We then conclude.

Then, there are two inductive cases. We extend the second \( i.h. \) to multi-types trivially.
Subcase $R = R'(u', z.r')$: Let consider the terms $t_0 = R'(D_n(\lambda x.s)(u,y,r))$ and $t_1 = R'((\{u/x\})D_n(s)/y)r$ so that $t = t_0(u', z.r')$ and $t' = t_1(u', z.r')$. The type derivation of $t'$ ends with a rule (APP) with the premises: $\Sigma_1 \vdash t_1 : \#([M_i \rightarrow \tau_i]_{i \in I})$, $\Delta \vdash u' : \#(\bigcup_{i \in I} M_i)$ and $z : [\tau_i]_{i \in I}; \Lambda \vdash r' : \sigma$, where $\Sigma = \Sigma_1 \uplus \Delta \uplus \Lambda$. By the second i.h. we get a derivation $\Gamma_0 \vdash t_0 : \#([M_i \rightarrow \tau_i]_{i \in I})$ for some $\Gamma_0$. We build a derivation for $t$ with type $\sigma$ ending with rule (APP) and using the derivations for $t_0$ and the ones for $u'$ and $r'$, so that the corresponding typing environment is $\Gamma = \Gamma_0 \uplus \Delta \uplus \Lambda$. We then conclude.

Subcase $R = n(u', z.R')$: Let $t_0, t_1$ be the same as before so that $t = n(u', z.t_0)$ and $t' = n(u', z.t_1)$. We detail the case where $z \in \text{fv}(t_0)$ and $z \notin \text{fv}(t_1)$, the other ones being similar to case 1. The type derivation of $t'$ is as follows, with $\Sigma = \Gamma_n \uplus \Delta \uplus \Sigma'$.

$$\frac{\Gamma_n \vdash n : [\tau] \quad \Delta \vdash u' : [\rho] \quad \Sigma' \vdash t_1 : \sigma}{\Gamma \vdash n(u', z.t_1) : \sigma} \text{(APP)}$$

By the second i.h. we have a derivation $z : [\tau_i]_{i \in I}; \Gamma' \vdash t_0 : \sigma$ for some $\Gamma'$. Also by relevance Lemma 4.1) we have $I \neq \emptyset$. By the i.h. on property $P$, we can build derivations $\Pi_i \vdash n : [] \rightarrow \tau_i$ for each $i \in I$ and thus a derivation $\Pi \vdash n : [] \rightarrow \tau_i_{i \in I}$ by rule (MANY) with $\Pi = \biguplus_{i \in I} \Pi_i$. Setting $\#(\bigcup_{i \in I} []) = [\rho]$, we then build the following derivation:

$$\frac{\Pi \vdash n : [] \rightarrow \tau_i_{i \in I} \quad \Delta \vdash u' : \#(\bigcup_{i \in I} []) \quad z : [\tau_i]_{i \in I}; \Gamma' \vdash t_0 : \sigma}{\Gamma \vdash n(u', z.t_0) : \sigma} \text{(APP)}$$

where $\Gamma = \Pi \uplus \Delta \uplus \Gamma'$. We thus conclude. $\square$

We finally obtain:

**Theorem 4.16 (Characterization).** System $\cap J$ characterizes strong normalization, i.e. $t$ is $\cap J$-typable if and only if $t \in \text{SN}(d\beta)$. 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 Proposition 4.11, while completeness holds by Lemma 4.15. The bound is given by Lemma 4.10. $\square$

### 4.3. Quantitative Behavior of $\pi$.

We have mentioned already that $\pi$ is rejected by the quantitative type systems $\cap J$. Concretely, this happens in the critical case when $x \notin \text{fv}(r)$ and $y \in \text{fv}(r')$ in

$$t_0 = t(u, x.r)(u', y.r') \rightarrow_{\pi} t(u, x.r(u', y.r')) = t_1$$

**Example 4.17.** We take $t_1 = x(y, a.z)(w, b(b(b(c,c)))) \rightarrow_{\pi} x(y, a.z(w, b(b(b(c,c)))) = t_2$. Let $\rho_1 = [\sigma] \rightarrow \tau$ and $\rho_2 = [\sigma] \rightarrow [\tau] \rightarrow \tau$. For each $i \in \{1, 2\}$ let $\Delta_i = x : [\sigma_1]; y : [\sigma_2]; z : [\rho_i]$. Consider

$$\Psi = \frac{b : [\tau] \rightarrow \tau \vdash b : [\tau] \rightarrow \tau \quad b : [\tau] \vdash b : [\tau]}{c : [\tau] \vdash c : [\tau]}$$

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

$$\Phi_i = \frac{x : [\sigma_1] \vdash x : [\sigma_1] \quad y : [\sigma_2] \vdash y : [\sigma_2] \quad z : [\rho_i] \vdash z : [\rho_i]}{\Delta_i \vdash x(y, a.z) : \rho_i}$$
Then, for the term $t_1$, we have the following derivation:

$$\begin{array}{c}
\Phi_1 \quad \Phi_2 \\
\Delta_1 \uplus \Delta_2 \vdash x(y, a.z) : [\rho_1, \rho_2] \\
\Gamma_1 \vdash x(y, a.z)(w, b(b(b(c, c)))) : \tau \\
\Psi = w : [\sigma, \sigma] \parallel w : [\sigma, \sigma]
\end{array}$$

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:

$$\begin{array}{c}
\Phi = z : [\rho_1, \rho_2] \parallel z : [\rho_1, \rho_2] \\
\Gamma_2 \vdash z(w, b(b(b(c, c)))) : \tau \\
\Psi = w : [\sigma, \sigma] \parallel w : [\sigma, \sigma]
\end{array}$$

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 $\Gamma_1$ and $\Gamma_2$ resp. are not the same. Despite the fact that the step $t_1 \rightarrow_{\pi} t_2$ 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 $\pi$-reduction: by assigning sets to variables instead of multisets, $\Gamma_1$ and $\Gamma_2$ would be equal.

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

**Lemma 4.18** (Typing behavior of $\pi$-reduction). Let $\Gamma \vdash_{R^1} t_1 : \sigma$. If $t_1 = (u, x.r)(u', y.r') \rightarrow_{\pi} t_2 = t(u, x.r(u', y.r'))$, then there are $n_2$ and $\Sigma \subseteq \Gamma$ such that $\Sigma \vdash_{R^2} t_2 : \sigma$ with $n_1 \geq n_2$.

**Proof.** The derivation of $t_1$ ends with (APP), with $\Gamma = \Gamma' \uplus \Delta_u \uplus \Delta_r$ and $n_1 = 1 + n' + n'_u + n'_r$. The reduction relation $\rightarrow_{\beta, \pi}$ will be used later in section 6 to show equivalence between $\text{SN}(\text{d} \beta)$ and $\text{SN}(\beta, \pi)$.

There are two possibilities.

Case $I \neq \emptyset$: Then $\#([\mathcal{M}_i \to \tau_i]_{i \in I}) = [\mathcal{M}_i \to \tau_i]_{i \in I}$ and for each $i \in I$ there is one derivation of $t(u, x.r)$ having the following form:

$$\begin{array}{c}
\Gamma_i \vdash^{n_i} t : \#([\mathcal{N}_j \to \rho_j]_{j \in J_i}) \quad \Delta^i \vdash^{n_u} u : \#(\cup_{j \in J_i} \mathcal{N}_j) \quad \Lambda^i_r ; x : [\rho_j]_{j \in J_i} \parallel^{n_r} r : \mathcal{M}_i \to \tau_i \\
\Gamma_i \uplus \Delta^i \uplus \Lambda^i_r \vdash t(u, x.r) : \mathcal{M}_i \to \tau_i (\text{APP})
\end{array}$$

where $\Gamma' = \cup_{i \in I} \Gamma_i \uplus \Delta_u \uplus \Delta_r$ and $n' = \sum_{i \in I} n^i + n^u + n^r$. From $(\Lambda^i_r ; x : [\rho_j]_{j \in J_i} \parallel^{n_r} r : \mathcal{M}_i \to \tau_i)_{i \in I}$ we can construct a derivation $\Phi_r = \cup_{i \in I} \Lambda^i_r ; x : [\rho_j]_{j \in J_i} \parallel^{+} r : \mathcal{M}_i \to \tau_i_{i \in I}$ using rule (MANY), where $J = \cup_{i \in I} J_i$. We then construct the following derivation:

$$\begin{array}{c}
\Phi_r \quad \Delta^i_u \vdash^{n_u} u' : \#(\cup_{i \in I} \mathcal{M}_i) \quad \Lambda^i_r ; y : [\tau_i]_{i \in I} \parallel^{n_r} r' : \sigma \\
\Psi = \cup_{i \in I} \Lambda^i_r \uplus \Delta^i_u \uplus \Lambda^i_r ; x : [\rho_j]_{j \in J_i} \vdash r(u', y.r') : \sigma
\end{array}$$

We then build two derivations $\Gamma \vdash^{n_t} t : \#([\mathcal{N}_j \to \rho_j]_{j \in J})$ with $\Gamma \subseteq \cup_{i \in I} \Gamma_i$ and $n_t \leq \sum_{i \in I} n^i$ and $\Delta^i_u \vdash^{n_u} u : \#(\cup_{j \in J} \mathcal{N}_j)$ with $\Gamma \subseteq \cup_{i \in I} \Gamma_u$ and $n_u \leq \sum_{i \in I} n^u_i$ as follows:
• If \( x \in \text{fv}(r) \), then all the \( J_i \)'s, and thus also \( J \), are non-empty by relevance so that 
\(#(\{N \to \rho\}_j \mid j \in I) = [\{N \to \rho\}_j \mid j \in I] \). Also, 
\(#(\{N \to \rho\}_j \mid j \in I) = [\{N \to \rho\}_j \mid j \in I] \). We 
obtain the expected derivation for \( t \) by Lemma 4.2, with \( \Gamma_t = \psi_{i \in I}^{\Gamma_i} \). 
Now for \( u \), notice that for each \( i \in I \) we can have either 
\(#(\{N \to \rho\}_j \mid j \in I) = [\{N \to \rho\}_j \mid j \in I] \) or,
if all the \( N_j \)'s are empty, 
\(#(\{N \to \rho\}_j \mid j \in I) = [\sigma_i] \) for some \( \sigma_i \) derived by \( \Delta^t_u \| n^t_u \ u : [\sigma_i] \). 
Then, there are two possibilities.

(1) If \( \cup_{j \in I} N_j = \emptyset \), we take an arbitrary \( k \in I \) and let 
\(#(\{N \to \rho\}_j \mid j \in I) = [\sigma_i] \) so that we 
can give a derivation \( \Delta_u \| n^u_u \ u : [\sigma_k] \) with \( \Delta_u = \Delta^t_u \subseteq \psi_{i \in I} \Delta^i_u \) and \( n_u = n^k_u \leq +i_{\in I} n^i_u \).

(2) Otherwise, we have 
\(#(\{N \to \rho\}_j \mid j \in I) = [\{N \to \rho\}_j \mid j \in I] \). Let \( I' \) be the subset of \( I \) such that 
for each \( i \in I' \) we have \( \cup_{j \in I} N_j \neq [\emptyset] \) and \( J' = \psi_{i \in I'} J_i \). By Lem. 30 we build 
ad derivation \( \Delta_u \| n^u_u \ u : [\{N \to \rho\}_j \mid j \in I] \) such that \( \Delta_u = \psi_{i \in I'} \Delta^i_u \subseteq \psi_{i \in I} \Delta^i_u \) and \( n_u = +i_{\in I} n^i_u \).

Finally, we build the following derivation of size \( n_2 \).

\[
\begin{aligned}
\Gamma_t \|^{n_t} t & : \#(\{N \to \rho\}_j \mid j \in I) \quad \Delta_u \|^{n_u} u \ : \#(\cup_{j \in I} N_j) \\
\Sigma & \vdash t(u, x, r(u', y, r')) : \sigma
\end{aligned}
\]

We have \( \Sigma = \Gamma_t \cup \Delta_u \psi_{i \in I} \Delta^i_t \cup \Delta_u \psi_{i \in I} \Delta^i_r \subseteq \Gamma \) and \( n_2 = n_t + n_u + i_{\in I} n^i_t + n_u' + n_r' \leq n_1 \).

**Case** \( I = \emptyset \): Then there is some \( \tau \) such that 
\(#(\{M \to \tau\}_i \mid i \in I) = [\tau] \) and the derivation of 
\( t(u, x, r) \) ends as follows:

\[
\begin{aligned}
\Gamma_t \|^{n_t} t & : \#(\{N \to \rho\}_j \mid j \in I) \quad \Delta_u \|^{n_u} u \ : \#(\cup_{j \in I} N_j) \\
& \quad \Delta_r ; x : [\rho]_j \mid j \in I \|^{n_r} r : \tau \\
\Sigma & \vdash t(u, x, r(u', y, r')) : \sigma
\end{aligned}
\]

Using \( \Gamma' = \Gamma_t \cup \Delta_u \cup \Delta_r \) and \( n' = n_t + n_u + n_r \).

We construct the following derivation of size \( n_2' \):

\[
\begin{aligned}
\Gamma_t \|^{n_t} t & : \#(\{N \to \rho\}_j \mid j \in I) \quad \Delta_u \|^{n_u} u \ : \#(\cup_{j \in I} N_j) \\
& \quad \Delta_r ; x : [\rho]_j \mid j \in I \|^{n_r} r : \tau \\
\Sigma & \vdash t(u, x, r(u', y, r')) : \sigma
\end{aligned}
\]

We have \( \Sigma = \Gamma_t \cup \Delta_u \cup \Delta_r \cup \Delta_u \psi_{i \in I} \Delta^i_r \psi_{i \in I} \Delta^i_r \subseteq \Gamma \) and \( n_2' = n_t + n_u + n_r + n_u' + n_r' \leq n_1 \).

We have proved that reducts of typed terms are also typed. To show that typed terms 
terminate, we will show that the maximal length of reduction to normal form is bounded 
by the size of the type derivation, so finite. This is similar to what we have done for \( \rightarrow_{d, \beta} \).

We recall that for each \( t \in \text{SN}(\beta, \pi) \), \( ||t||_{\beta, \pi} \) represents the maximal length of a \( \beta, \pi \)-
reduction sequence to \( \beta, \pi\text{-nf} \) starting at \( t \). We also define \( ||t||_{\beta, \pi} \) as the maximal number 
of \( \beta \)-steps in \( \beta, \pi \)-reduction sequences from \( t \) to \( \beta, \pi \)-normal form. Notice that, in general,
Lemma 4.19. If \( t_1 \to_\beta t_2 \) and \( t_1 \to_\pi t_3 \), then there is \( t_4 \) such that \( t_3 \to_\beta t_4 \) and \( t_2 \to_\pi t_4 \).

Proof. By case analysis of the possible overlaps of the two contracted redexes.

Lemma 4.20. If \( t_1 \to_\beta t_2 \), then there is \( t_3 \) such that \( \pi(t_1) \to_\beta t_3 \) and \( t_2 \to_\pi t_3 \).

Proof. By induction on the reduction sequence from \( t_1 \) to \( \pi(t_1) \) using Lemma 4.19 for the base case.

Lemma 4.21. If there is a \( \beta, \pi \)-reduction sequence \( \rho \) starting at \( t \) and containing \( k \) \( \beta \)-steps, then there is a \( \beta, \pi \)-reduction sequence \( \rho' \) starting at \( \pi(t) \) and also containing \( k \) \( \beta \)-steps.

Proof. By induction on the (necessarily finite) reduction sequence \( \rho \). If the length of \( \rho \) is 0, then \( k = 0 \) and the property is trivial. If the length of \( \rho \) is \( 1 + n \), we analyze the two possible cases:

1. If \( \rho \) is \( t \to_\beta t' \) followed by \( \rho_0 \) of length \( n \) and containing \( k_0 = k - 1 \) \( \beta \)-steps, then the property holds for \( t' \) w.r.t. \( \pi(t') \). But Lemma 4.20 gives a term \( t'' \) such that \( \pi(t) \to_\beta t'' \) and \( t' \to_\pi t'' \). Then we construct the \( \beta, \pi \)-reduction sequence \( \pi(t) \to_\beta t'' \to_\pi \pi(t'') = \pi(t') \) followed by the one obtained by the i.h. This new sequence has \( 1 + k_0 = k \) \( \beta \)-steps.

2. If \( \rho \) is \( t \to_\pi t' \) followed by \( \rho_0 \) of length \( n \) and containing \( k_0 = k \) \( \beta \)-steps, then the property holds for \( t' \) w.r.t. \( \pi(t') \). Since \( \pi(t) = \pi(t') \), we are done by the i.h.

Lemma 4.22. \( ||t||^\beta_{\beta, \pi} = ||\pi(t)||^\beta_{\beta, \pi} \).

Proof. First we prove \( ||t||^\beta_{\beta, \pi} \leq ||\pi(t)||^\beta_{\beta, \pi} \). If there is a \( \beta, \pi \)-reduction sequence starting at \( t \) and containing \( k \) \( \beta \)-steps, then the same happens for \( \pi(t) \) by Lemma 4.21. Next we prove \( ||t||^\beta_{\beta, \pi} \geq ||\pi(t)||^\beta_{\beta, \pi} \). If there is a \( \beta, \pi \)-reduction sequence starting at \( \pi(t) \) and containing \( k \) \( \beta \)-steps, then the same happens for \( t \) because it is sufficient to prefix this sequence with the steps \( t \to_\pi \pi(t) \). We conclude \( ||t||^\beta_{\beta, \pi} = ||\pi(t)||^\beta_{\beta, \pi} \).

Lemma 4.23. If \( t \to_\pi t' \), then \( ||t||^\beta_{\beta, \pi} = ||t'||^\beta_{\beta, \pi} \).

Proof. We have \( ||t||^\beta_{\beta, \pi} = ||\pi(t)||^\beta_{\beta, \pi} = ||\pi(t'||^\beta_{\beta, \pi} = ||t'||^\beta_{\beta, \pi} \).

Lemma 4.24. The following equalities hold:

| Expression | Equality |
|------------|----------|
| \( ||x||^\beta_{\beta, \pi} \) | 0 |
| \( ||\lambda x.t||^\beta_{\beta, \pi} \) | \( ||t||^\beta_{\beta, \pi} \) |
| \( ||x(u, y.r)||^\beta_{\beta, \pi} \) | \( ||u||^\beta_{\beta, \pi} + ||r||^\beta_{\beta, \pi} \) |
| \( ||(\lambda x.t)(u, y.r)||^\beta_{\beta, \pi} \) | \( 1 + ||\{(u/x)t/y\}r||^\beta_{\beta, \pi} \) if \( x \in \text{fv}(t) \) and \( y \in \text{fv}(r) \) |
| \( ||(\lambda x.t)(u, y.r)||^\beta_{\beta, \pi} \) | \( 1 + ||\{(u/x)t/y\}r||^\beta_{\beta, \pi} + ||u||^\beta_{\beta, \pi} \) if \( x \notin \text{fv}(t) \) and \( y \in \text{fv}(r) \) |
| \( ||(\lambda x.t)(u, y.r)||^\beta_{\beta, \pi} \) | \( 1 + ||r||^\beta_{\beta, \pi} + ||t||^\beta_{\beta, \pi} + ||u||^\beta_{\beta, \pi} \) if \( y \notin \text{fv}(r) \) |
| \( ||t(u, x.r)(u', y.r')||^\beta_{\beta, \pi} \) | \( ||t(u, x.r(u', y.r'))||^\beta_{\beta, \pi} \) |

Proof. The proof follows from the inductive definition ISN(\( \beta, \pi \)) and Lemma 4.23.

Lemma 4.25. If \( \Gamma \vdash^k \tau : \sigma \), then \( ||t||^\beta_{\beta, \pi} \leq k \).
Proof. We define $|x|_l = |\lambda x.t|_l = 0, |t(u,x.r)|_l = |t|_l + 1$. We proceed by induction on the pair $(k, |t|_l)$ with respect to the lexicographic order and we reason by case analysis on $t$. The proofs for cases $t = x$, $t = \lambda x.u$, $t = x(u,x.r)$ and $t = (\lambda x.s)(u,y,r)$ are similar to the ones in Lemma 4.10, only replacing $||t||_{d\beta}$ by $||t||_{\beta,\pi}^\beta$. We only show here the most interesting case which is $t = s(u,x.r)(u',y,r')$.

Let $t' = s(u,x.r(u',y,r'))$. By Lemma 4.18 there is a type derivation $\Delta \vdash t' : \sigma$ with $k' \leq k$ and $\Delta \subseteq \Gamma$. Since $||t||_{\beta,\pi} = $Lemma 4.24 $||t'||_{\beta,\pi}^\beta$ and $|t|_l > |t'|_l$ we can use the i.h. and we get $||t'||_{\beta,\pi}^\beta \leq k'$, and thus $||t||_{\beta,\pi}^\beta \leq k' \leq k$.

As a corollary we obtain:

**Lemma 4.26 (Soundness for $\Lambda J$).** If $t$ is $\cap J$-typable, then $t \in SN(\beta, \pi)$.

**Proof.** By Lemma 4.25, the number of $\beta$-reduction steps in any $\beta, \pi$-reduction sequence starting at $t$ is finite. So in any infinite $\beta, \pi$-reduction sequence starting at $t$, there is necessarily a term $u$ from which there is an infinite amount of $\pi$-steps only. But this is impossible since $\pi$ terminates, so we conclude by contradiction.

---

5. Faithfulness of the Translation

The original translation of generalized applications into ES (see [ES07]), based on $t(u,x.r)^* = \{t^*u^*/x\}r^*$, is not conservative with respect to strong normalization; this is also true for the original translation to $\lambda$-terms given by [JM03], which is based on $t(u,x.r)^* = \{t^*u^*/x\}r^*$: it preserves strong normalization but normalizes too much. Indeed, in a $\beta$-redex $s := (\lambda x.t_0)(u,y,r)$, the interaction of $\lambda x.t_0$ with the argument $u$ is materialized by the internal substitution in the contractum term $\{(u/x)t_0/\delta\}r$. Such interaction may be elusive: if the external substitution is vacuous (that is, if $y$ is not free in $r$), $\beta$-reduction will simply throw away the $\lambda$-abstraction $\lambda x.t_0$ and its argument $u$. In the translated term $s^*$, the $\beta$-redex $(\lambda x.t_0)^*u^* = (\lambda x.t_0)^*u$ is also thrown away in the case of translation to $\lambda$-terms, whereas it may reduce in the context of the explicit substitution $[(\lambda x.t_0)^*u^*/y]r^*$.

The different interactions between the abstraction and its argument in the two mentioned models of computation has important consequences. Here is an example.

**Example 5.1.** Let $\delta := \lambda x.x(x, z, z)$. Let $r$ be a $T_J$-term with no free occurrences of $y$, e.g. $r = \lambda y.y$. The only possible reduction from the $T_J$-term $\delta(\delta, y.r)$ is to $r = \lambda x.x$, which is a normal form in $\Lambda J$ or $\Lambda J_n$, whereas the subterm $\delta^*\delta^* = (\lambda x.[x/x/\delta^*]z)(\lambda x.[x/\delta^*]z)$ may reduce forever in the context of a vacuous explicit substitution, i.e. $[\delta^*\delta^*/y]r^* \rightarrow^+ [\delta^*\delta^*/y]r^*$ holds in the ES calculus.

In this section we define an alternative encoding to the original one and prove it faithful: a term in $T_J$ is $d\beta$-strongly normalizing iff its alternative encoding is strongly normalizing in the ES framework. In a later section, we use this connection with ES to establish the equivalence between strong normalization of $\Lambda J_n$ and $\Lambda J$. 
5.1. A New Translation. We define the syntax and semantics of an ES calculus borrowed from \[\text{Acc12}\] to which we relate \(\lambda J_n\). It is a simple calculus where \(\beta\) 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 \(\langle[N/x](\lambda y.M)\rangle\) \(\rightarrow_{\text{dB}} [N/x][P/y]M\), where the ES \([N/x]\) lies between the abstraction and its argument. Terms and list contexts are given by:

\[
\begin{align*}
(T_{\lambda ES}) & \quad M, N, P, Q := x \mid \lambda x.M \mid MN \mid [N/x]M \\
(\text{List contexts}) & \quad L := \odot \mid [N/x]L
\end{align*}
\]

The calculus \(\lambda ES\) is defined by \(T_{\lambda ES}[\text{dB}, \text{sub}]\), meaning that \(T_{\lambda ES}\) is the set of terms and that this set is equipped with \(\rightarrow_{\text{dB}}\) and \(\rightarrow_{\text{sub}}\), the reduction relations obtained by closing dB, sub under all contexts, where:

\[
L(\lambda x.M)N \rightarrow_{\text{dB}} L([N/x]M) \\
[N/x]M \rightarrow_{\text{sub}} \{N/x\}M
\]

Now, consider the (original) translation from \(T_{J}\) to \(T_{\lambda ES}\) [ES07]:

\[
x^* := x \quad (\lambda x.t)^* := \lambda x.t^* \\
t(u, y.r)^* := [t^*u^*/y]r^*
\]

According to it, the notion of distance in \(\lambda ES\) corresponds to our notion of distance for \(\lambda J_n\). For instance, the application \(t(u, x_\ldots)\) in the term \(t(u, x_\lambda y.(u', z.r'))\) can be seen as a substitution \([t^*u^*/x]\) inserted between the abstraction \(\lambda y.r\) and the argument \(u'\). But how can we now (informally) relate \(\pi\) to the notions of existing permutations for \(\lambda ES\)? Using the previous translation, we can see that \(t_0 = t(u, x.\lambda r(u', y.r')) \rightarrow_{\pi} t(u, x.\lambda r(u', y.r')) = t_1\) simulates as

\[
t_0^* = [([t^*u^*/x]r^*)u^*/y]r'^* \rightarrow [[t^*u^*/x](r^*u'^*)/y]r'^* \rightarrow [t^*u^*/x][r^*u'^*/y]r'^* = t_1^*. 
\]

The first step is an instance of a rule in ES known as \(\sigma_1:\) \(\langle[u/x]t\rangle v \rightarrow [u/x](tv)\), and the second one of a rule we call \(\sigma_4:\) \([u/x]t/y]v \rightarrow [u/x][y/t]v\). Quantitative types for ES tell us that only rule \(\sigma_1\), but not rule \(\sigma_4\), is valid for a call-by-name calculus. This is why it is not surprising that \(\pi\) is rejected by our type system, as detailed in subsection 4.3.

The alternative encoding we propose is as follows (noted \((\cdot)^*\) instead of \((\cdot)^{**}\)):

**Definition 5.2** (Translation from \(T_{J}\) to \(T_{\lambda ES}\)).

\[
x^* := x \quad (\lambda x.t)^* := \lambda x.t^* \\
t(u, x.\lambda r)^* := [t^*/x'][u^*/x']\{x^1x^2/x\}r^*
\]

where \(x^1\) and \(x^2\) are fresh variables.

Notice the above \(\pi\)-reduction \(t_0 \rightarrow t_1\) is still simulated: \(t_0^* \rightarrow_{\sigma_4}^2 t_1^*\). Moreover, consider again the counterexample \(t = \delta(\delta, y.r)\) to faithfulness (Example 5.1). The alternative encoding of \(t\) is now giving by \([\delta^*/y^1][\delta^*/y^1]^*\{y^1y^2/y\}r^*\), which is just \([\delta^*/y^1][\delta^*/y^1]^*r^*\), because \(y \notin \text{fv}(r^*)\). The only hope to have an interaction between the two copies of \(\delta^*\) in the previous term is to execute the ES, but such executions will just throw away those two copies, because \(y^1, y^2 \notin \text{fv}(r^*)\). This hopefully 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 \( \lambda J_n \) and the one of its encoding in \( \lambda ES \). 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 \( \lambda ES \), we will use the following system [KV20]:

**Definition 5.3 (The Type System \( \cap ES \)).**

\[
\begin{align*}
&x: [\sigma] \vdash x: \sigma \quad \text{(AX)} \\
&\frac{\psi \in i, i \vdash M : \sigma_i}{\Gamma \vdash \prod_{i \in I} M : [\sigma_i]_{i \in I}} \quad \text{(MANY)} \\
&\frac{\Gamma \vdash M : \mathcal{M} \rightarrow \sigma}{\Gamma \vdash \lambda x. M : \mathcal{M} \rightarrow \sigma} \quad \text{(\( \rightarrow_i \))}
\end{align*}
\]

**Theorem 5.4.** Let \( M \in T_{ES} \). Then \( M \) is typable in \( \cap ES \) iff \( M \in SN(dB, \text{sub}) \).

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

**Lemma 5.5.** Let \( t \in T_I \). Then \( \Gamma \models_{\cap I} t : \sigma \implies \Gamma \models_{\cap ES} t^* : \sigma \).

**Proof.** By induction on the type derivation. Notice that the statement also applies by straightforward \( i.h. \) for rule (MANY).

**Case (VAR):** Then \( t = x \) and we type \( t^* = x \) with rule (VAR).

**Case (ABS):** Then \( t = \lambda x. s \) and \( t^* = \lambda x. s^* \). We conclude by \( i.h. \) using (\( \rightarrow_i \)).

**Case (APP):** Then \( t = s(u, x, r) \) and \( t^* = [s^*/x^1][u^*/x^2][x^1 x^2/x^3] r^* \). By the \( i.h. \), we have derivations \( \Pi \models_{\cap ES} s^* : \#(\mathcal{M}_i \rightarrow \tau_i)_{i \in I} \), \( \Delta \models_{\cap ES} u^* : \#(\cup_{i \in I} \mathcal{M}_i) \) and \( \Lambda; x : [\tau_i]_{i \in I} \). If \( I \neq \emptyset \), it is easy to construct a derivation \( \frac{\Phi}{\Pi; \Delta; x^1 : [\mathcal{M}_i \rightarrow \tau_i]_{i \in I} : \{x^1 x^2/x\} r^* \sigma} \quad \Pi \models u^* : [\mathcal{M}_i \rightarrow \tau_i]_{i \in I} \). We conclude by building the following derivation.

\[
\begin{align*}
&\Phi \\
&\frac{\Pi; \Delta; x^1 : [\mathcal{M}_i \rightarrow \tau_i]_{i \in I} : \{x^1 x^2/x\} r^* \sigma}{\Pi; \Delta; x : [\cup_{i \in I} \mathcal{M}_i] \vdash \#(\mathcal{M}_i) \vdash [\mathcal{M}_i \rightarrow \tau_i]_{i \in I}}
\end{align*}
\]

If \( I = \emptyset \), then \( x \notin \text{fv}(r) = \text{fv}(r^*) \) by relevance, so that \( t^* = [s^*/x^1][u^*/x^2] r^* \). By the \( i.h. \), we have derivations \( \Pi \models_{\cap ES} s^* : \tau \), \( \Delta \models_{\cap ES} u^* : [\rho] \) and \( \Pi \models r^* : \sigma \) with \( \Gamma = \Pi \vdash \Delta \vdash \Lambda \). We conclude by building the following derivation.

\[
\begin{align*}
&\frac{\Pi; \Delta; x^1 : [\mathcal{M}_i \rightarrow \tau_i]_{i \in I} : \{x^1 x^2/x\} r^* \sigma}{\Pi; \Delta; x^1 : [\mathcal{M}_i \rightarrow \tau_i]_{i \in I} : \{x^1 x^2/x\} r^* \sigma}
\end{align*}
\]

This last result, together with the two characterization Theorem 4.16 and Theorem 5.4, gives:

**Corollary 5.6.** Let \( t \in T_J \). If \( t \in SN(dB) \) then \( t^* \in SN(dB, \text{sub}) \).

We show the converse by a detour through the encoding of \( T_{ES} \) to \( T_J \).
**Definition 5.7** (Translation from $T_{ES}$ to $T_J$).

\[
x^o := x \\
(MN)^o := M^o(N^o, x.x) \\
(\lambda x.M)^o := \lambda x.M^o \\
(M[N/x])^o := I(N^o, x.M^o)
\]

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

**Lemma 5.8.** Let $M \in T_{ES}$. Then $\Gamma \triangleright_{\cap ES} M : \sigma \implies \Gamma \triangleright_{\cap J} M^o : \sigma$.

*Proof.* By induction on the derivation. The cases where the derivation ends with (VAR), (ABS) or (MANY) (generalizing the statement) are straightforward.

**Case (APP):** Then $M = PN$ and $M^o = P^o(N^o, z.z)$. By the i.h., we have derivations $\Lambda \triangleright_{\cap J} P^o : \mathcal{M} \rightarrow \sigma$ and $\Delta \triangleright_{\cap J} N^o : \#(\mathcal{M})$ with $\Gamma = \Lambda \triangleright \Delta$. By application of rule (MANY) we obtain $\Lambda \triangleright_{\cap J} P^o : [\mathcal{M} \rightarrow \sigma]$. We conclude by building the following derivation.

\[
\frac{\Lambda \triangleright P^o : [\mathcal{M} \rightarrow \sigma]}{} \quad \frac{\Delta \triangleright N^o : \#(\mathcal{M})}{\Delta \triangleright P^o(N^o, x.x) : \sigma} \quad \frac{x : [\sigma] \triangleright x : \sigma}{\Delta \triangleright P^o(N^o, x.x) : \sigma}
\]

**Case (ES):** Then $M = P[x/N]$ and we have a translation of the form $M^o = (\lambda z.z)(N^o, x.P^o)$. By the i.h., we have derivations $\Lambda; x : \mathcal{M} \triangleright_{\cap J} P^o : \sigma$ and $\Delta \triangleright_{\cap J} N^o : \#(\mathcal{M})$ with $\Gamma = \Lambda \triangleright \Delta$. Let $\mathcal{M} = \{[\tau_i]_{i \in I}\}$.

If $I \neq \emptyset$, we conclude by building the following derivation.

\[
\frac{\emptyset \triangleright \lambda z.z : [\tau_i] \rightarrow [\tau_i]}{} \quad \frac{\Delta \triangleright N^o : \#(\mathcal{M})}{\Delta \triangleright (\lambda z.z)(N^o, x.P^o) : \sigma} \quad \frac{\Lambda; x : \mathcal{M} \triangleright P^o : \sigma}{\Delta \triangleright (\lambda z.z)(N^o, x.P^o) : \sigma}
\]

If $I = \emptyset$, we conclude by building the following derivation (where $\tau$ is arbitrary).

\[
\frac{\emptyset \triangleright \lambda z.z : [\tau] \rightarrow [\tau]}{} \quad \frac{\Delta \triangleright N^o : \#(\mathcal{M})}{\Delta \triangleright (\lambda z.z)(N^o, x.P^o) : \sigma} \quad \frac{\Lambda; x : \mathcal{M} \triangleright P^o : \sigma}{\Delta \triangleright (\lambda z.z)(N^o, x.P^o) : \sigma}
\]

**Lemma 5.9.** Let $t \in T_J$. Then $\Gamma \triangleright_{\cap J} t^* : \sigma \implies \Gamma \triangleright_{\cap J} t : \sigma$.

*Proof.* By induction on $t$. The cases where $t = x$ or $t = \lambda x.s$ are straightforward by the i.h. We reason by cases for the generalized application.

**Case** $t = s(u, x.r)$ where $x \in \text{fv}(r)$: We have

\[
t^* = (s^*/x^1)[u^* / x^1][x^1/x^1]r^*)^o = I(s^*, x^1, I(u^*, x^1, \{x^1(x^1, z.z) / x\}r^*)
\]

By construction and also by the anti-substitution Lemma 4.13 it is not difficult to see that $\Gamma = \Gamma_\triangleright \triangleright \cap J r^* \triangleright$ and there exist derivations having the following conclusions, where $I \neq \emptyset$:

1. $\Gamma; r : [\tau_i]_{i \in I} \triangleright_{\cap J} r^* : \sigma$
2. $x^1 : [[\tau_i] \rightarrow [\tau_i]]_{i \in I} \triangleright_{\cap J} x^1 : [[\tau_i] \rightarrow [\tau_i]]_{i \in I}$
3. $x^1 : [[\tau_i]_{i \in I} \triangleright_{\cap J} x^1 : [\tau_i]_{i \in I}$


Theorem 5.11 (Faithfulness) in the previous sections. From it, we obtain new results about the original calculus $\Lambda$.

Case $t$ provides the main result of this section:

In this section we compare the various concepts of strong normalization that are induced in the previous section, we related strong $\beta$-normalization with strong normalization of ES.

Corollary 5.10. Let $t \in T_J$. Then $\Gamma \vdash_{\cap J} t : \sigma \iff \Gamma \vdash_{\cap J} t^* : \sigma$.

Equivalent Notions of Strong Normalization

In the previous section, we related strong $d\beta$-normalization with strong normalization of ES. In this section we compare the various concepts of strong normalization that are induced on $T_J$ by $\beta$, $d\beta$, $(\beta, p2)$ and $(\beta, \pi)$. This comparison makes use of several results obtained in the previous sections. From it, we obtain new results about the original calculus $\Lambda_J$. 

6. Equivalent Notions of Strong Normalization
6.1. \(\beta\)-Normalization is not Enough. Obviously, \(\text{SN}(d\beta) \subseteq \text{SN}(\beta)\), since \(\beta \subseteq d\beta\). Similarly, \(\text{SN}(\beta, \pi) \subseteq \text{SN}(\beta)\) and \(\text{SN}(\beta, p2) \subseteq \text{SN}(\beta)\). We now see that these inclusions are strict. We have discussed in subsection 2.2 the unblocking property of \(\pi\) and \(p2\) and the unblocked character of distant redexes. From the point of view of normalization, this means that \(T_J[\beta]\) has premature normal forms and that \(\text{SN}(\beta) \nsubseteq \text{SN}(d\beta)\); similarly for the other inclusions above. To illustrate this we give an example of a \(T_J\)-term which normalizes when only using rule \(\beta\), but diverges when adding permutation rules or distance. Let us take \(t := w(u, u', \delta)(\delta, x, x)\), where \(\delta\) is the term of Example 5.1. Although this term is a normal form in \(T_J[\beta]\), the second \(\delta\) is actually an argument for the first one, as we can see with a \(\pi\) permutation:

\[
t \rightarrow_{\pi} w(u, u', \delta(\delta, x, x)) := t' \]

Thus \(t \rightarrow_{\pi} t' \rightarrow_{\beta} t'\) which implies \(t \notin \text{SN}(\beta, \pi)\). We can also unblock the redex in \(t\) by a \(p2\)-permutation moving the inner \(\lambda x\) up:

\[
t \rightarrow_{p2} (\lambda y. w(u, u', y(y, z, z)))(\delta, x, x) \rightarrow_{\beta} t' \]

Thus \(t \rightarrow_{p2} t' \rightarrow_{\beta} t'\) and thus \(t \notin \text{SN}(\beta, p2)\). Finally, we get the same thing in a unique \(d\beta\)-step: \(t \rightarrow_{d\beta} t' \rightarrow_{d\beta} t'\).

In all the three cases, \(\beta\)-strong normalization is not preserved by the permutation rules, as there is a term \(t \in \text{SN}(\beta)\) such that \(t \notin \text{SN}(\beta, \pi)\), \(t \notin \text{SN}(\beta, p2)\) and \(t \notin \text{SN}(d\beta)\).

6.2. Comparison with \(\beta + p2\). We now formalize the fact that our calculus \(T_J[d\beta]\) is a version with distance of \(T_J[\beta, p2]\), so that they are equivalent from a normalization point of view. For this, we will establish the equivalence between strong normalization w.r.t. \(d\beta\) and \(\beta, p2\), through a long chain of equivalences. One of them is Theorem 5.11, that we have proved in the previous section; the other is a result about \(\sigma\)-rules in the \(\lambda\)-calculus – which is why we have to go through the \(\lambda\)-calculus again.

**Definition 6.1** (Translation \(\downarrow\) from \(T_{ES}\) to \(T_{\Lambda}\)).

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

**Lemma 6.2.** Let \(M \in T_{ES}\). Then \(M \in \text{SN}(d\beta, \text{sub}) \implies M^\downarrow \in \text{SN}(\beta)\).

**Proof.** For typability in the \(\lambda\)-calculus, we use the type system \(\Sigma_{\lambda}\) with choice operators of [KV20]. It can be seen as a restriction of the system \(\cap ES\) to \(\lambda\)-terms. Suppose \(M \in \text{SN}(d\beta, \text{sub})\). By Theorem 5.4 \(M\) is typable in \(\cap ES\), and it is straightforward to show that \(M^\downarrow\) is typable in \(\Sigma_{\lambda}\). Moreover, \(M^\downarrow\) typable implies that \(M^\downarrow \in \text{SN}(\beta)\) ([KV20]), which is what we want. \(\square\)

For \(t \in T_J\), let \(t^\square := (t^*)^\downarrow\). So, we are just composing the alternative encoding of generalized application into \(ES\) with the map into \(\lambda\)-calculus just introduced. The translation \((\cdot)^\downarrow\) may be given directly by recursion as follows:

\[
x^\square = x \quad (\lambda x. t)^\square = \lambda x. t^\square \quad t(u, y, r)^\square = (\lambda y'. (\lambda y. \{y' \mapsto y\}) r^\square) u^\square
\]

**Lemma 6.3.** \(t^\square \in \text{SN}(\beta, \sigma_2) \implies t \in \text{SN}(\beta, p2)\).

**Proof.** Because \((\cdot)^\sqcup\) produces a strict simulation from \(T_J\) to \(T_{\Lambda}\). More precisely: (i) if \(t_1 \rightarrow_{\beta} t_2\) then \(t_1^\square \rightarrow_{\beta}^\sigma t_2^\square\); (ii) if \(t_1 \rightarrow_{p2} t_2\) then \(t_1^\square \rightarrow_{\sigma_2}^2 t_2^\square\). \(\square\)

**Theorem 6.4.** Let \(t \in T_J\). Then \(t \in \text{SN}(\beta, p2)\) iff \(t \in \text{SN}(d\beta)\).
Proof. We prove that the following conditions are equivalent: 1) \( t \in SN(\beta, p^2) \). 2) \( t \in SN(d\beta) \). 3) \( t^* \in SN(dB, sub) \). 4) \( t^\beta \in SN(\beta) \). 5) \( t^\beta \in SN(\beta, \sigma^2) \). Now, 1) \( \Rightarrow \) 2) is because \( \rightarrow_{d\beta} \subset \rightarrow_{\beta, p^2}^+ \). 2) \( \Rightarrow \) 3) is by Corollary 5.6. 3) \( \Rightarrow \) 4) is by Lemma 6.2. 4) \( \Rightarrow \) 5) is showed by [Reg94]. 5) \( \Rightarrow \) 1) is by Lemma 6.3. 

Incidentally, the previous proof also contains a new proof of Theorem 5.11.

6.3. Comparison with \( \beta + \pi \). We now prove the equivalence between strong normalization for \( d\beta \) and for \( (\beta, \pi) \). One of the implications already follows from the properties of the typing system.

Lemma 6.5. Let \( t \in T_J \). If \( t \in SN(d\beta) \) then \( t \in SN(\beta, \pi) \).

Proof. Follows from the completeness of the typing system (Lemma 4.15) and soundness of \( \cap J \) for \( (\beta, \pi) \) (Lemma 4.26).

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 \( \Lambda J \)-calculus; 3) Two new properties of strong normalization for \( \beta, \pi \) in \( \Lambda J \); and 4) Preservation of strong \( \beta, \pi \)-normalization by a certain map from the set \( T_J \) into itself.

The remark about explicit substitutions is this:

Lemma 6.6. For all \( M \in T_{ES} \), \( M \in SN(dB, sub) \) iff \( M \in SN(B, sub) \).

The translation \( (\cdot)^\circ \) in Definition 5.7 induces a simulation of each reduction step \( \rightarrow_{sub} \) on \( T_{ES} \) into a reduction step \( \rightarrow_\beta \) on \( T_J \), but cannot simulate the creation of an ES effected by rule dB. A solution is to refine the translation \( (\cdot)^\circ \) for applications, yielding the following alternative translation:

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

Since the clause for ES is not changed, simulation of each reduction step \( \rightarrow_{sub} \) by a reduction step \( \rightarrow_\beta \) holds as before. The improvement lies in the simulation of each dB-reduction step:

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

This strict simulation gives immediately:

Lemma 6.7. For all \( M \in T_{ES} \), if \( M^\bullet \in SN(\beta) \) then \( M \in SN(B, sub) \).

We now prove two properties of strong normalization for \( (\beta, \pi) \) in \( \Lambda J \). Following [Mat00], \( SN(\beta, \pi) \) admits an inductive characterization \( ISN(\beta, \pi) \), given in Figure 1, which uses the following inductive generation for \( T_J \)-terms:

\[
t, u, r := xS \mid \lambda x.t \mid (\lambda x.t)S\overline{S} \quad S := (u, y, r)
\]

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

A preliminary fact is the following:
\[ x \in \text{ISN}(\beta, \pi) \quad \text{(VAR)} \]
\[ u, r \in \text{ISN}(\beta, \pi) \quad \frac{x(u, z) \in \text{ISN}(\beta, \pi)}{x(u, z)(y)/x \in \text{ISN}(\beta, \pi)} \quad \text{(HVAR)} \]
\[ t \in \text{ISN}(\beta, \pi) \quad \frac{\lambda x.t \in \text{ISN}(\beta, \pi)}{\lambda x.t(u, z) \in \text{ISN}(\beta, \pi)} \quad \text{(LAMBDA)} \]
\[ x(u, y)S^S \in \text{ISN}(\beta, \pi) \quad \frac{x(u, y)S \in \text{ISN}(\beta, \pi)}{x(u, y)S^S \in \text{ISN}(\beta, \pi)} \quad \text{(PI)} \]
\[ \{u/x\}t/y \in \text{ISN}(\beta, \pi) \quad \frac{\{\{u/x\}t/y\}r^S \in \text{ISN}(\beta, \pi)}{t, u \in \text{ISN}(\beta, \pi)} \quad \text{(BETA)} \]

Figure 1: Inductive characterization of the strong \( \beta, \pi \)-normalizing \( \Lambda J \)-terms

**Lemma 6.8.** The set \( \text{SN}(\beta, \pi) \) is closed under prefixing of arbitrary \( \pi \)-reduction steps:
\[
\frac{t \rightarrow^* \pi t' \text{ and } t' \in \text{SN}(\beta, \pi)}{t \in \text{SN}(\beta, \pi)}
\]

**Proof.** We first consider the following three facts:

1. Every \( t \in T_J \) has a unique \( \pi \)-normal form \( \pi(t) \).
2. The map \( \pi(\cdot) \) preserves \( \beta \)-reduction steps, that is, \( t_1 \rightarrow^* \beta t_2 \) implies \( \pi(t_1) \rightarrow^* \beta \pi(t_2) \) (Lemma 4.20).
3. \( \rightarrow^* \pi \) is terminating.

Now, suppose \( t \not\in \text{SN}(\beta, \pi) \), so that there is an infinite \( (\beta, \pi) \)-reduction sequence starting at \( t \). Then by the previous facts it is possible to construct an infinite \( \beta \)-reduction sequence starting at \( \pi(t) \). But \( \pi(t) = \pi(t') \) and \( t' \rightarrow^* \pi(t') \), so there is an infinite \( \beta, \pi \)-reduction sequence starting at \( t' \), which leads to a contradiction. \( \square \)

Given that \( \text{SN}(\beta, \pi) = \text{ISN}(\beta, \pi) \), the “rule” in Lemma 6.8, when written with \( \text{ISN}(\beta, \pi) \), is admissible for the predicate \( \text{ISN}(\beta, \pi) \). Now, consider:
\[
\frac{u, r \in \text{ISN}(\beta, \pi)}{\{y(u, z)\}r^S \in \text{ISN}(\beta, \pi)} \quad \text{(I)}
\]
\[
\frac{\{\{u/y\}t/z\}r^S \in \text{ISN}(\beta, \pi) \quad t, u \in \text{ISN}(\beta, \pi) \quad x \notin \text{fv}(t, u, r)}{(\lambda x.t)(u, z) \in \text{ISN}(\beta, \pi)} \quad \text{(II)}
\]

Notice rule (II) generalizes rule (beta): just take \( r = xS^S \), with \( x \not\in \tilde{S} \).

The two new properties of strong normalization for \( \beta, \pi \) in \( \Lambda J \) are contained in the following lemma.

**Lemma 6.9.** Rules (I) and (II) are admissible rules for the predicate \( \text{ISN}(\beta, \pi) \).

**Proof.** Proof of (I). By induction on \( t \in \text{ISN}(\beta, \pi) \), we prove that \( \{y(u, z)\}t \in \text{ISN}(\beta, \pi) \).

The most interesting case is (p1), which we spell out in detail. We will use a device to shorten the writing: if \( E \) is \( t \), or \( S \), or \( \tilde{S} \), then \( E \) denotes \( \{y(u, z)\}t \). Suppose \( t = y'(u', z')S^S \in \text{ISN}(\beta, \pi) \) with \( y'(u', z')S \tilde{S} \in \text{ISN}(\beta, \pi) \). We want \( t \in \text{ISN}(\beta, \pi) \). If \( y' \neq y \), then the thesis follows by the i.h. and one application of (p1). Otherwise, \( t = y(u, z)(u', z')S^S \). By the i.h.,
\[
y(u, z)(u', z')S^S \in \text{ISN}(\beta, \pi).
\]

By inversion of (p1), we get
\[
y(u, z)(u', z')S^S \in \text{ISN}(\beta, \pi).
\]
From this, Lemma 6.8 gives
\[ y(u, z, z'(u, z')) \in \text{ISN}(\beta, \pi). \]

Finally, two applications of (pt) yield \( t \in \text{ISN}(\beta, \pi) \).

Proof of (II). We prove the following: for all \( t_1 \in \text{ISN}(\beta, \pi) \), for all \( n \geq 0 \), if \( t_1 \) has \( n \) occurrences of the sub-term \( \{\{u/y\}t/z\}r \), then, for any choice of \( n \) such occurrences, \( t_2 \in \text{ISN}(\beta, \pi) \), where \( t_2 \) is the term that results from \( t_1 \) by replacing each of those \( n \) occurrences by \( (\lambda y.t)(u, z, r) \).

Notice the statement we are going to prove entails the admissibility of (II). Indeed, given \( s \), let \( n \) be the number of free occurrences of \( x \) in \( s \). The term \( t_1 = \{\{u/y\}t/z\}r \) has well determined \( n \) occurrences of the sub-term \( \{\{u/y\}t/z\}r \) (it may have others), and \( \{\langle \lambda y.t\rangle(u, z, r)/x\}s \) is the term that results from \( t_1 \) by replacing each of those \( n \) occurrences by \( (\lambda y.t)(u, z, r) \).

Suppose \( t_1 \in \text{ISN}(\beta, \pi) \) and consider \( n \) occurrences of the sub-term \( \{\{u/y\}t/z\}r \) in \( t_1 \). The proof is by induction on \( t_1 \in \text{ISN}(\beta, \pi) \) and sub-induction on \( n \). A term \( s \) is determined, with \( n \) free occurrences of \( x \), such that \( x \notin t, u, r \) and \( t_1 = \{\{u/y\}t/z\}r/x \) \( s \). We want to prove that \( \{\langle \lambda y.t\rangle(u, z, r)/x\}s \in \text{ISN}(\beta, \pi) \). We will use a device to shorten the writing: if \( E \) is \( t, \) or \( S \), or \( \bar{S} \), then \( E \) denotes \( \{\{u/y\}t/z\}r/x \) \( E \) and \( \bar{E} \) denotes \( \{\langle \lambda y.t\rangle(u, z, r)/x\}E \). The proof proceeds by case analysis on \( s \).

We show the critical case \( s = x\bar{S} \), where use is made of the sub-induction hypothesis. We are given \( s = \{\{u/y\}t/z\}r\bar{S} \in \text{ISN}(\beta, \pi) \). We want to show \( s = \{\langle \lambda y.t\rangle(u, z, r)\}r\bar{S} \in \text{ISN}(\beta, \pi) \). Given that \( t, u \in \text{ISN}(\beta, \pi) \), it suffices
\[ \{\{u/y\}t/z\}r\bar{S} \in \text{ISN}(\beta, \pi) \] due to invertibility of (beta). Let \( s' = \{\{u/y\}t/z\}r\bar{S} \). Since \( x \notin t, u, r \), we have \( s' = s \) (whence \( s' \in \text{ISN}(\beta, \pi) \)), and the number of free occurrences of \( x \) in \( s' \) is \( n - 1 \). By sub-induction hypothesis, \( s' \in \text{ISN}(\beta, \pi) \). But \( s' = \{\{u/y\}t/z\}r\bar{S} \), again due to \( x \notin t, u, r \). Therefore Equation 6.1 holds.

We now move to the fourth part of the ongoing reasoning. Consider the map from \( T_J \) to itself obtained by composing \( (\beta)^* : T_J \to T_{ES} \) with \( (\beta)^* : T_{ES} \to T_J \). Let us write \( t^i \) this composition. A recursive definition is also possible, as follows:
\[ x^i = x \quad \lambda x.t^i = \lambda x.t^i \quad t^i(u, y, r) = I(t^i, y_1, I(u^i, y_2, \{y_1(x_2, z, z)/y\}r^i)) \]

**Lemma 6.10.** If \( t \in \text{SN}(\beta, \pi) \) then \( t^i \in \text{SN}(\beta, \pi) \).

**Proof.** For \( t \in \text{SN}(\beta, \pi) \), \( ||t||_{\beta, \pi} \) denotes the length of the longest \( \beta, \pi \)-reduction sequence starting at \( t \). We prove \( t^i \in \text{SN}(\beta, \pi) \) by induction on the longest \( \beta, \pi \) reduction sequence starting at \( t \) (\( ||t||_{\beta, \pi} \)), with sub-induction on the size of \( t \). We proceed by case analysis of \( t \).

**Case** \( t = x \): We have \( x^i = x \in \text{SN}(\beta, \pi) \).

**Case** \( t = \lambda x.s \): We have \( t^i = \lambda x.s^i \). The sub-inductive hypothesis gives \( s^i \in \text{SN}(\beta, \pi) \). By rule (LAMBDA), \( \lambda x.s^i \in \text{SN}(\beta, \pi) \).

**Case** \( t = y(u, x, r) \): We have \( t^i = I(y(x_1, x_2, \{x_1(x_2, z, z)/x\}r^i)) \). By the (sub)-i.h., \( u^i, r^i \in \text{SN}(\beta, \pi) \). Rule (I) yields \( \{y(u^i, z, z)/x\}r^i \in \text{SN}(\beta, \pi) \). Applying rule (BETA) twice, we obtain \( t^i \in \text{SN}(\beta, \pi) \).

**Case** \( t = (\lambda y.s)(u, x, r) \): We have \( t^i = I(\lambda y.s^i, x_1, I(u^i, x_2, \{x_1(x_2, z, z)/x\}r^i)) \). Notice that \( ||t^i||_{\beta, \pi} \) is greater than \( ||s^i||_{\beta, \pi} \) and \( ||u^i||_{\beta, \pi} \). The induction hypothesis, \( s^i, u^i \in \text{SN}(\beta, \pi) \). Also \( ||t^i||_{\beta, \pi} > ||\{\{u/y\}s/x\}r||_{\beta, \pi} \). Hence \( \{\{u/y\}s/x\}r \in \text{SN}(\beta, \pi) \), again
by the i.h. Since map $(\cdot)^\dagger$ commutes with substitution, \(\{u^\dagger/y\}s^\dagger/x) \in \text{ISN}(\beta, \pi)\).

This, together with \(s^\dagger \in \text{ISN}(\beta, \pi)\), gives \(((\lambda y.s^\dagger)(u^\dagger, z.z)/x) \in \text{ISN}(\beta, \pi)\), due to rule (II). Applying rule (\text{beta}) twice, we obtain \(t^\dagger \in \text{ISN}(\beta, \pi)\).

**Case** \(t = t_0(u_1, x.r_1)(u_2, y.r_2)\): Let \(s := t_0(u_1, x.r_1(u_2, y.r_2))\). Since \(t \to_\pi s\), the i.h. gives \(s^\dagger \in \text{ISN}(\beta, \pi)\). The induction hypothesis also gives \(t_0^\dagger, u_1^\dagger \in \text{ISN}(\beta, \pi)\). The term \(s^\dagger\) is

\[
I(t_0^\dagger, x_1.I(u_1^\dagger, x_2.\{x_1(x_2, z.z)/x\}I(r_1^\dagger, y_1.I(u_2^\dagger, y_2.\{y_1(y_2, z.z)/y\}r_2^\dagger)))
\]

From \(s^\dagger \in \text{ISN}(\beta, \pi)\), by four applications of (\text{beta}) we obtain

\[
\{(\{t_0^\dagger(u_1^\dagger, z.z)/x\}r_1^\dagger)(u_2^\dagger, z.z)/y\}r_2^\dagger \in \text{ISN}(\beta, \pi) \quad (6.2)
\]

We want \(t^\dagger \in \text{ISN}(\beta, \pi)\), where \(t^\dagger\) is

\[
I(I(t_0^\dagger, x_1.I(u_1^\dagger, x_2.\{x_1(x_2, z.z)/x\}r_1^\dagger)), y_1.I(u_2^\dagger, y_2.\{y_1(y_2, z.z)/y\}r_2^\dagger)))
\]

From Equation 6.2 and \(u_1^\dagger \in \text{ISN}(\beta, \pi)\), rule (II) obtains

\[
\{I(u_1^\dagger, x_2.\{t_0^\dagger(x_2, z.z)/x\}r_1^\dagger(u_2^\dagger, z.z)/y\}r_2^\dagger \in \text{ISN}(\beta, \pi)
\]

From this, Lemma 6.8 (prefixing of \(\pi\)-reduction steps) obtains

\[
\{I(u_1^\dagger, x_2.\{t_0^\dagger(x_2, z.z)/x\}r_1^\dagger)(u_2^\dagger, z.z)/y\}r_2^\dagger \in \text{ISN}(\beta, \pi)
\]

From this and \(t_0^\dagger \in \text{ISN}(\beta, \pi)\), rule (II) obtains

\[
\{I(t_0^\dagger, x_1.I(u_1^\dagger, x_2.\{x_1(x_2, z.z)/x\}r_1^\dagger)(u_2^\dagger, z.z)/y\}r_2^\dagger \in \text{ISN}(\beta, \pi)
\]

From this, Lemma 6.8 (prefixing of \(\pi\)-reduction steps) obtains

\[
\{I(t_0^\dagger, x_1.I(u_1^\dagger, x_2.\{x_1(x_2, z.z)/x\}r_1^\dagger))(u_2^\dagger, z.z)/y\}r_2^\dagger \in \text{ISN}(\beta, \pi)
\]

Finally, two applications of (\text{beta}) obtain \(t^\dagger \in \text{ISN}(\beta, \pi) = \text{SN}(\beta, \pi)\).

All is in place to obtain the desired result:

**Theorem 6.11.** Let \(t \in T_I\). Then \(t \in \text{SN}(d\beta)\) iff \(t \in \text{SN}(\beta, \pi)\).

**Proof.** The implication from left to right is Lemma 6.5. For the converse, suppose \(t \in \text{SN}(\beta, \pi)\). By Lemma 6.10, \(t^\dagger \in \text{SN}(\beta, \pi)\). Trivially, \(t^\dagger \in \text{SN}(\beta)\). Since \(t^\dagger = (t^*)^\dagger\), Lemma 6.7 gives \(t^* \in \text{SN}(\beta, \sub)\). By Lemma 6.6, \(t^* \in \text{SN}(d\beta, \sub)\). By Theorem 5.11, \(t \in \text{SN}(d\beta)\). \(\Box\)

6.4. **Consequences for \(\Lambda J\).** Our previous results for \(\lambda J_n\) provide new ones for the original \(\Lambda J\) (a quantitative typing system characterizing strong normalization, and a faithful translation into ES) as immediate consequences of Theorems 4.16, 5.11 and 6.11.

**Theorem 6.12.** Let \(t \in T_I\). Then:

1. **(Characterization)** \(t \in \text{SN}(\beta, \pi)\) iff \(t\) is \(\cap J\)-typable.
2. **(Faithfulness)** \(t \in \text{SN}(\beta, \pi)\) iff \(t^* \in \text{SN}(d\beta, \sub)\).

Beyond strong normalization, \(\Lambda J\) gains a new normalizing strategy, which reuses the notion of left-right normal form introduced in subsection 3.2. We take the definitions of neutral terms, answer and left-right context \(R\) given there for \(\lambda J_n\), in order to define a new left-right strategy and a new predicate \(\text{ISNj}\) for \(\Lambda J\). The strategy is defined as the
closure under \( R \) of rule \( \beta \) and of the particular case of rule \( \pi \) where the redex has the form \( n(u, x.a)S \).³

**Definition 6.13.** Predicate \( ISNj \) is defined by the rules \( (snvar) \), \( (snapp) \), \( (snabs) \) in Definition 3.7, together with the following two rules (which replace rule \( (snbeta) \)):

\[
\begin{align*}
R(n(u, y.a)S) & \in ISNj \\
R(n(u, y.a)S') & \in ISNj
\end{align*}
\]

\( (snredef1) \) \hspace{1cm} \( (snredef2) \)

The corresponding normalization strategy is organized as usual: an initial phase obtains a left-right normal form, whose components are then reduced by internal reduction. Is this new strategy any good? Theorem 6.16 answers positively with the equivalence between \( ISNj \) and \( ISN(\beta, \pi) \). Before proving it, we need two intermediate lemmas.

**Lemma 6.14.** The following rule is admissible for the predicate \( ISNj \):

\[
\begin{align*}
u, r & \in ISNj \\
x(u, y.r) & \in ISNj
\end{align*}
\]

**Proof.** The proof is by induction on \( r \in ISNj \). If \( r \) is generated by rules \( (snvar) \), \( (snapp) \) or \( (snabs) \), then \( r \) is a weak-head normal form and rule \( (snapp) \) applies. Otherwise \( r = R(redex) \). By inversion of rules \( (snredef1) \) and \( (snredef2) \), one obtains \( R(contractum) \) in \( ISNj \), plus two other subterms of the redex also in \( ISNj \) in case of \( (snredef1) \). Let \( R' := x(u, y.R) \). By the \( i.h. \) \( R'(contractum) \) in \( ISNj \). By one of the rules \( (snredef1)/(snredef2) \), \( R'(redex) \) in \( ISNj \), that is \( x(u, y.r) \in ISNj \).

**Lemma 6.15.** The following rule is admissible for the predicate \( ISNj \):

\[
\begin{align*}
n(u, y.sS)S & \in ISNj \\
n(u, y.s)SS & \in ISNj
\end{align*}
\]

**Proof.** We prove by induction on \( r \in ISNj \), that, if \( r = n(u, y.sS)S \), then \( n(u, y.s)SS \in ISNj \). We do case analysis of \( s \).

**Case** \( s = a \): Follows by rule \( (snredef1) \) by taking \( R = aS \).

**Case** \( s = R(redex) \): Let \( R_1 := n(u, y.RS)S \) and \( R_2 := n(u, y.R)SS \). Since \( r = R_1(redex) \), inversion of rule \( (snredef1)/(snredef2) \) gives \( R_1(contractum) \) in \( ISNj \), plus two other subterms of the redex also in \( ISNj \) in case of \( (snredef2) \). By \( i.h. \) \( R_2(contractum) \) in \( ISNj \). A final application of \( (snredef1)/(snredef2) \) gives \( R_2(redex) \) in \( ISNj \), as required.

**Case** \( s = n' \): First, notice there are exactly four sub-cases:

**Subcase** \( n'S \) is a weak-head normal form and \( S \) is empty: By inversion of \( (snapp) \), we take \( sS \) apart, obtain its components in \( ISNj \) and, using \( (snapp) \), we reconstruct the term \( n(u, y.n'S) \) in \( ISNj \).

**Subcase** \( S \) has the form \( (u', y'.R(redex)) \) and \( S \) is arbitrary: By inversion of the rule \( (snredef1)/(snredef2) \), we have \( n(u, y.n'(u', y'.R(contractum)))S \in ISNj \), plus two other subterms of the redex also in \( ISNj \) in case of \( (snredef2) \). By the \( i.h. \), we have that \( n(u, y.n')(u', y'.R(contractum))S \in ISNj \). As required, we obtain \( n(u, y.n')(u', y'.R(redex))S \in ISNj \) by rule \( (snredef1)/(snredef2) \).

³Notice how a redex has the two possible forms \( (\lambda x.t)S \) or \( n(u, x.a)S \), that can be written as \( aS \), that is, the form \( \beta_n(\lambda x.t)S \) of a left-right redex in \( \lambda J_n \).
Subcase \( S \) has the form \((u', y'.a)\) and \( \vec{S} \) is non-empty: Let \( \vec{S} = R \vec{R} \). By applying inversion of \((\text{SNREDEX1})\) twice, we obtain \( n(u, y.n'(u', y'.aR)) \vec{R} \in \text{ISN}_j \). By \( \text{i.h.} \), \( n(u, y.n')(u', y'.aR) \vec{R} \in \text{ISN}_j \). By \((\text{SNREDEX1})\), \( n(u, y.n')(u', y'.aR) \vec{R} \in \text{ISN}_j \), as required.

Subcase \( S \) has the form \((u', y', n'')\) and \( \vec{S} \) is non-empty: We have to analyze \( \vec{S} \). For that, we introduce some notation. \( R^\text{ml} \) (respectively \( R^\text{ans}, R^\text{whnf}, R^\text{rdx} \)) will denote a generalized argument of the form \((t, z.n)\) (resp. \((t, z.a), (t, z.w)\text{whnf}, (t, z.R(\text{redex}))\)).

Let \( n_0 = n(u, y.n'(u', y'.n'')) \) and \( n_1 = n(u, y.n')(u', y'.n''). \) The non-empty \( \vec{S} \) has exactly 3 possible forms (in all cases \( m \geq 0 \)).

Subsubcase \( R^\text{ml}_1 \cdots R^\text{ml}_m R^\text{whnf} \): We apply the same kind of reasoning as in subcase 1.

Subsubcase \( R^\text{ml}_1 \cdots R^\text{ml}_m R^\text{rdx} \vec{R} \): Let \( R^\text{rdx} = (u'', y''.R''(\text{redex})) \) and let

\[
R_0 = n_0 R^\text{ml}_1 \cdots R^\text{ml}_m (u'', y''.R'') \vec{R} \\
R_1 = n_1 R^\text{ml}_1 \cdots R^\text{ml}_m (u'', y''.R'') \vec{R}
\]

Inversion of rule \((\text{SNREDEX1})/(\text{SNREDEX2})\) gives \( R_0 \langle \text{contractum} \rangle \in \text{ISN}_j \), plus two other subterms of the redex also in \( \text{ISN}_j \) in case of \((\text{SNREDEX2})\). By the \( \text{i.h.} \), we have that \( R_1 \langle \text{contractum} \rangle \in \text{ISN}_j \). We obtain \( R_1 \langle \text{redex} \rangle \in \text{ISN}_j \) by rule \((\text{SNREDEX1})/(\text{SNREDEX2})\), as required.

Subsubcase \( R^\text{ml}_1 \cdots R^\text{ml}_m R^\text{ans}_{m+1} R^\text{rdx} \vec{R} \): Let \( R^\text{ans}_{m+1} = (u'', y'.a) \) and let

\[
n_2 = n_0 R^\text{ml}_1 \cdots R^\text{ml}_m \\
n_3 = n_1 R^\text{ml}_1 \cdots R^\text{ml}_m
\]

By inversion of \((\text{SNREDEX1})\), we obtain \( n_2(u'', y''.aR_{m+2}) \vec{R} \in \text{ISN}_j \). Next \( \text{i.h.} \) gives \( n_3(u'', y''.aR_{m+2}) \vec{R} \in \text{ISN}_j \). By \((\text{SNREDEX1})\), \( n_3(u'', y''.a)R_{m+2} \vec{R} \in \text{ISN}_j \), as required.

**Theorem 6.16.** Let \( t \in T_J \). Then \( t \in \text{ISN}_j \) iff \( t \in \text{ISN}((\beta, \pi)) \).

**Proof.** \( \Rightarrow \) We show that each rule defining \( \text{ISN}_j \) is admissible for the predicate \( \text{ISN}((\beta, \pi)) \) defined in Figure 1. Cases \((\text{SNVAR})\) and \((\text{SNAbs})\) are straightforward. Case \((\text{SNREDEX1})\) is by the \( \text{i.h.} \) and Lemma 6.8. Case \((\text{SNREDEX2})\) is by the \( \text{i.h.} \) and rule (II). Case \((\text{SNAPP})\) is proved by a straightforward induction on \( n \).

\( \Leftarrow \) We show that each rule in Figure 1 defining the predicate \( \text{ISN}((\beta, \pi)) \) is admissible for the predicate \( \text{ISN}_j \). Cases \((\text{VAR})\) and \((\text{LAMBDAA})\) are straightforward. Case \((\text{BETA})\) is by rule \((\text{SNREDEX2})\) and the \( \text{i.h.} \), by just taking \( R = o \vec{S} \). Case \((\text{HVAR})\) follows by Lemma 6.14 and the \( \text{i.h.} \). Case \((\text{pi})\) is by Lemma 6.15 and the \( \text{i.h.} \). \( \square \)

6.5. **Alternative Proof of Equivalence.** The last theorem can also be shown as a corollary of \( \text{ISN}_j = \text{SN}((\beta, \pi)) \) and the fact that \( \text{SN}((\beta, \pi)) = \text{ISN}((\beta, \pi)) \) proved by [JM03]. We will show the first equality \( \text{ISN}_j = \text{SN}((\beta, \pi)) \) in a similar way as for \( d\beta \) (Theorem 3.11).

**Lemma 6.17.** If \( t_0 \rightarrow^{\beta, \pi} t_1 \), then

- \( \{u/x\} t_0 \rightarrow^{\beta, \pi} \{u/x\} t_1 \), and
- \( \{t_0/x\} u \rightarrow^{\beta, \pi} \{t_1/x\} u \).
Proof. The first statement is proved by induction on $t_0 \rightarrow_{\beta,\pi} t_1$ using Lemma 2.7. The second is proved by induction on $u$.

Lemma 6.18. The strategy introduced in subsection 6.4 is deterministic.

Proof. For every term there is a unique decomposition in terms of a $R$ context and a redex. Besides that, $\beta$ and $\pi$ redexes do not overlap.

Lemma 6.19. If $t_0 = R(\langle u/x \rangle t/y) \in SN(\beta, \pi)$ and $t, u \in SN(\beta, \pi)$, then $t'_0 = R(\langle \lambda x.t \rangle(u, y)) \in SN(\beta, \pi)$.

Proof. By hypothesis we also have $r \in SN(\beta, \pi)$. We use the lexicographic order to reason by induction on $\langle ||t_0||_{\beta,\pi}, \langle |t|_{\beta,\pi}, |u|_{\beta,\pi}, R \rangle$. To show $t'_0 \in SN(\beta, \pi)$ it is sufficient to show that all its reducts are in $SN(\beta, \pi)$. We analyze all possible cases.

Case: $t'_0 \rightarrow_{\beta,\pi} t_0$. We conclude by the hypothesis.

Case: $t'_0 \rightarrow_{\beta,\pi} R(\langle \lambda x.t' \rangle(u, y)) = t'_1$, where $t \rightarrow_{\beta,\pi} t'$. We have $t', u \in SN(\beta, \pi)$ and by item (2) $t_0 = R(\langle \{u/x\} t/y \rangle) \rightarrow_{\beta,\pi} R(\langle \{u/x\} t'/y \rangle) = t_1$, so that also $t_1 \in SN(\beta, \pi)$. We conclude $t'_1 \in SN(\beta, \pi)$ by the i.h. since $||t_1||_{\beta,\pi} < ||t_0||_{\beta,\pi}$ and $||t'_1||_{\beta,\pi} < ||t||_{\beta,\pi}$.

Case: $t'_0 \rightarrow_{\beta,\pi} R(\langle \lambda x.t \rangle(u', y)) = t'_1$, where $u \rightarrow_{\beta,\pi} u'$. We have $t, u' \in SN(\beta, \pi)$ and by item (2) $t_0 = R(\langle \{u/x\} t/y \rangle) \rightarrow_{\beta,\pi} R(\langle \{u'/x\} t/y \rangle) = t_1$, so that also $t_1 \in SN(\beta, \pi)$. We conclude $t'_1 \in SN(\beta, \pi)$ by the i.h. since $||t_1||_{\beta,\pi} < ||t_0||_{\beta,\pi}$ and $||u'||_{\beta,\pi} < ||u||_{\beta,\pi}$.

Case: $t'_0 \rightarrow_{\beta,\pi} R(\langle \lambda x.t \rangle(u, y r)) = t'_1$, where $r \rightarrow_{\beta,\pi} r'$. We have $t, u \in SN(\beta, \pi)$ and by item (1) $t_0 = R(\langle \{u/x\} t/y \rangle) \rightarrow_{\beta,\pi} R(\langle \{u/x\} t/r \rangle) = t_1$, so that also $t_1 \in SN(\beta, \pi)$. We conclude $t'_1 \in SN(\beta, \pi)$ by the i.h. since $||t_1||_{\beta,\pi} < ||t_0||_{\beta,\pi}$.

Case: $t'_0 \rightarrow_{\beta,\pi} R' \langle \langle \lambda x.t \rangle(u, y) \rangle S = t'_1$, where $R \rightarrow_{\beta,\pi} R'$. Thus we also have that $t_0 = R(\langle \{u/x\} t/y \rangle) \rightarrow_{\beta,\pi} R'(\langle \{u/x\} t/y \rangle r) = t_1$. We have $t, u \in SN(\beta, \pi)$. We conclude that $t'_1 \in SN(\beta, \pi)$ by the i.h. since $||t_1||_{\beta,\pi} < ||t_0||_{\beta,\pi}$.

Case: $R = R'(\langle \circ S \rangle)$ and $t'_0 = R'(\langle \lambda x.t \rangle(u, y) r) \rightarrow_{\beta,\pi} R'(\langle \lambda x.t \rangle(u, y) r S) = t'_1$. This is the only case left. We have $t_0 = R'(\langle \{u/x\} t/y \rangle r S) = R'(\langle \{u/x\} t/y \rangle r S) = t_1$. We also have $t, u \in SN(\beta, \pi)$. We conclude $t'_1 \in SN(\beta, \pi)$ by the i.h. on $R$ since $\langle ||t_1||_{\beta,\pi}, |t||_{\beta,\pi}, |u||_{\beta,\pi} \rangle = \langle ||t_0||_{\beta,\pi}, |t||_{\beta,\pi}, |u||_{\beta,\pi} \rangle$. Notice that when $R = \circ$, then $\pi$-reduction can only take place in some subterm of $t'_0$, already considered in the previous cases.

Lemma 6.20. If $t_0 = R(\langle u, y, a \rangle S) \in SN(\beta, \pi)$, then $t'_0 = R(\langle u, y, a \rangle S) \in SN(\beta, \pi)$.

Proof. We use the lexicographic order to reason by induction on $\langle ||t_0||_{\beta,\pi}, n \rangle$. To show $t'_0 \in SN(\beta, \pi)$ it is sufficient to show that all its reducts are in $SN(\beta, \pi)$. We analyze all possible cases.

Case: $t'_0 \rightarrow_{\pi} t_0$: We conclude by the hypothesis.

Case: $t'_0 \rightarrow_{\beta,\pi} R(\langle u, y, a \rangle S) = t'_1$, where $n \rightarrow_{\beta,\pi} n'$: We have $t_0 \rightarrow_{\beta,\pi} R(\langle n' \rangle(u, y, a \rangle S)) = t_1$, so that also $t_1 \in SN(\beta, \pi)$. We conclude $t'_1 \in SN(\beta, \pi)$ by the i.h. since $||t_1||_{\beta,\pi} < ||t_0||_{\beta,\pi}$.

Case: $t'_0 \rightarrow_{\beta,\pi} R(\langle u, y, a \rangle S) = t'_1$, where $u \rightarrow_{\beta,\pi} u'$: We have $t_0 \rightarrow_{\beta,\pi} R(\langle n(u', y, a \rangle S)) = t_1$, so that also $t_1 \in SN(\beta, \pi)$. We conclude $t'_1 \in SN(\beta, \pi)$ by the i.h. since $||t_1||_{\beta,\pi} < ||t_0||_{\beta,\pi}$.

Case: $t'_0 \rightarrow_{\beta,\pi} R(\langle u, y, a \rangle S) = t'_1$, where $a \rightarrow_{\beta,\pi} a'$: We have $t_0 \rightarrow_{\beta,\pi} R(\langle n(u, y, a' \rangle S)) = t_1$, so that also $t_1 \in SN(\beta, \pi)$. We conclude $t'_1 \in SN(\beta, \pi)$ by the i.h. since $||t_1||_{\beta,\pi} < ||t_0||_{\beta,\pi}$.

Case: $t'_0 \rightarrow_{\beta,\pi} R(\langle u, y, a \rangle S') = t'_1$, where $S \rightarrow_{\beta,\pi} S'$: We have $t_0 \rightarrow_{\beta,\pi} R(\langle n(u, y, a \rangle S')) = t_1$, so that also $t_1 \in SN(\beta, \pi)$. We conclude $t'_1 \in SN(\beta, \pi)$ by the i.h. since $||t_1||_{\beta,\pi} < ||t_0||_{\beta,\pi}$.
Case $R = R'(\alpha S')$: Thus, $t'_0 = R'\langle n(u, y, a)(u', z, r)S' \rangle \rightarrow_{\pi} R'\langle n(u, y, a)(u', z, r)S' \rangle = t'_1$, where $S = (u', z, r)$. Then, $t_0 = R'\langle n(u, y, a)(u', z, r)S' \rangle \rightarrow_{\pi} R'\langle n(u, y, a)(u', z, r)S' \rangle = t_1$, so that also $t_1 \in SN(\beta, \pi)$. We conclude $t'_1 \in SN(\beta, \pi)$ by the i.h. since $||t_1||_{\beta, \pi} < ||t_0||_{\beta, \pi}$.

Case $n = n''(u', z, n')$: Thus $t'_0 = R'\langle n''(u', z, n')(u, y, a)S \rangle \rightarrow_{\pi} R'\langle n''(u', z, n')(u, y, a)S \rangle = t'_1$.

We do a case analysis on all the one-step reducts of $t'_0$ so we need to consider $t'_1$ with $S$ outside. We have $t_0 \rightarrow_{\pi} R'\langle n''(u', z, n')(u, y, a)S \rangle = t_1$, so that also $t_1 \in SN(\beta, \pi)$. Let $R' = R(n''(u', z, a))$. We have $||t_1||_{\beta, \pi} < ||t_0||_{\beta, \pi}$ so by the i.h. $R'\langle n''(u, y, a)S \rangle \in SN(\beta, \pi)$. Because $n''(u, y, a)$ is an answer we can apply the i.h. on $n''$ and we conclude $t'_1 \in SN(\beta, \pi)$.

\textbf{Lemma 6.21.} $\text{ISN}_j = SN(\beta, \pi)$.

\textbf{Proof.} First, we show $\text{ISN}_j \subseteq SN(\beta, \pi)$. We proceed by induction on $t \in \text{ISN}_j$.

\textbf{Case t = x: } Straightforward.

\textbf{Case t = λx.s, where s ∈ ISNj: } By the i.h. $s \in SN(\beta, \pi)$, so that $t \in SN(\beta, \pi)$ trivially holds.

\textbf{Case t = n(u, x, r) where n, u, r ∈ ISNj and r ∈ NF}_h: Since n is stable by reduction, n cannot in particular reduce to an answer. Therefore any kind of reduction starting at t only occurs in the subterms n, u and r. We conclude since n, u, r \in SN(\beta, \pi) hold by the i.h.

\textbf{Case t = R(n(u, y, a)S), where R(n(u, y, a)S) ∈ ISNj: } The i.h. gives $R(n(u, y, a)S) \in SN(\beta, \pi)$, so that $t \in SN(\beta, \pi)$ holds by Lemma 6.20.

\textbf{Case t = R(λx.s)(u, y, r), where R(\{u/x\}s/y)r, s, u ∈ ISNj: } The i.h. gives $R(\{u/x\}s/y)r \in SN(\beta, \pi)$, $s \in SN(\beta, \pi)$ and $u \in SN(\beta, \pi)$ so that $t \in SN(\beta, \pi)$ holds by Lemma 6.19.

Next, we show $SN(\beta, \pi) \subseteq ISNj$. Let $t \in SN(\beta, \pi)$. We reason by induction on $\langle||t||_{\beta, \pi}, |t|\rangle$ w.r.t. the lexicographic order. If $\langle||t||_{\beta, \pi}, |t|\rangle$ is minimal, i.e. $\langle0, 1\rangle$, then t is a variable and thus in ISNj by rule (SNVAR). Otherwise we proceed by case analysis.

\textbf{Case t = λx.s: } Since $||s||_{\beta, \pi} = ||t||_{\beta, \pi}$ and $|s| < |t|$, we conclude by the i.h. and rule (SNABS).

\textbf{Case t is an application: } There are three cases.

\textbf{Subcase t ∈ NF}_h: Then $t = n(u, x, r)$ with n, u, r \in SN(\beta, \pi) and r \in NF_h. We have $||n||_{\beta} \leq ||t||_{\beta, \pi}$, $||u||_{\beta, \pi} \leq ||t||_{\beta, \pi}$, $||r||_{\beta, \pi} \leq ||t||_{\beta, \pi}$, $|n| < |t|$, $|u| < |t|$ and $|r| < |t|$.

By the i.h. n, u, r \in ISNj and thus we conclude by rule (SNAPP).

\textbf{Subcase t = R(λx.s)(u, y, r): } $t \in SN(\beta, \pi)$ implies in particular $R(\{u/x\}s/y)r, s, u \in SN(\beta, \pi)$, so that they are in ISNj by the i.h. We conclude that $t \in ISNj$ by rule (SNREDX2).

\textbf{Subcase t = R(n(u, y, a)S): } $t \in SN(\beta, \pi)$ implies in particular $R(n(u, y, a)S) \in SN(\beta, \pi)$, so that this term is in ISNj by the i.h. We conclude $t \in ISNj$ by rule (SNREDX1).

\textbf{7. Conclusion}

\textbf{Contributions.} This paper presents and studies several properties of the call-by-name $\lambda J_n$-calculus, a formalism implementing an appropriate notion of distant reduction to unblock the $\beta$-redexes arising in generalized application notation.

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

We have also defined a faithful translation from the $\lambda_{Jn}$-calculus into ES. The translation preserves strong normalization, in contrast to the traditional translation from generalized applications to ES e.g. in [ES07]. Last but not least, we related strong normalization of $\lambda_{Jn}$ with that of other calculi, including in particular the original $\Lambda J$ of Joachimski and Matthes [JM03, JM00]. New results for the latter were found by means of the techniques developed for $\lambda_{Jn}$. In particular, a quantitative characterization of strong normalization was developed for $\Lambda J$, where the bound of reduction given by the size of type derivations only holds for $\beta$-steps (and not for $\pi$-steps).

This paper is an extended version of [ESKP22]. In this version we provide full proofs, and improve the presentation and discussion. The proof of confluence for $\lambda_{Jn}$ given in subsection 2.3 comes from [Pey22].

Related work. Generalizing elimination rules of natural deduction is an old idea, occurring several times in the literature, most notably by [SH84b, SH84a] or [Ten92, Ten02], before being coined in the version at the origin of $\Lambda J$ by von Plato [vP01]. The generalization of implication elimination itself has come up independently along the years, as pointed out by [SH14].

Concerning $\Lambda J$, some interesting results were given, motivated by a proof-theoretical approach. In parallel to his works with Joachimski [JM00, JM03] introducing the calculus, [Mat01] proves an interpolation theorem (with information on terms) for $\Lambda J$ extended with pairs and sum datatypes. In his PhD thesis, [Bar08] defines a set of conversions for $\Lambda J$ beyond $\beta$ and $\pi$. Some of these conversions where already given by [Mat01], another one is an undirected version of p2. Espírito Santo and his coauthors have used $\Lambda J$, and his military extension $\Lambda J_m$ [ESP03] to compare the computational content of natural deduction and the sequent calculus [ES09, ESFP16]. The call-by-value variant of $\Lambda J$ was introduced in [ES20].

The first non-idempotent type system for generalized applications was proposed in our conference paper [ESKP22]. Intersection type systems for $\Lambda J$ have been given before in [Mat00] and [ESIL12], but these systems handle idempotent types, so that they are not able to characterize quantitative properties. Since [ESKP22], further investigations on generalized applications based on distant reduction appeared in [KP22, Pey22]. Other calculi based on different logical systems have been adapted to enable quantitative analyzes: this is for instance the case of $\lambda\mu$ based on classical logic [KV20], or the Curry-Howard interpretation of the intuitionistic sequent calculus $\bar{\lambda}$ [KV15].

Future work. Quantitative type systems, introduced here for the call-by-name system $\lambda_{Jn}$, have been successfully adapted to the call-by-value setting in [KP22]. Further unification between call-by-name and call-by-value with the help of generalized applications could be considered in the setting of call-by-push-value [Lev06] or the polarized lambda-calculus [ES16].

It would be interesting to see if the techniques developed for tightness [AGLK20, KV22] can be adapted to this framework. The precise measures on reduction length obtained would enable us to precisely measure the quantitative relationship between the call-by-name $\lambda$-calculus and $\lambda_{Jn}$. Such techniques could also be adopted for call-by-value, to sharpen the relation between generalized applications and call-by-value calculi.
References

[Acc12] Beniamino Accattoli. An abstract factorization theorem for explicit substitutions. In 23rd International Conference on Rewriting Techniques and Applications (RTA), volume 15 of Leibniz International Proceedings in Informatics (LIPIcs), pages 6–21, Dagstuhl, Germany, 2012. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.RTA.2012.6.

[AGLK20] Beniamino Accattoli, Stéphane Graham-Lengrand, and Delia Kesner. Tight typings and split bounds, fully developed. Journal of Functional Programming, 30(ICFP), 2020. doi:10.1017/s095679682000012x.

[AK10] Beniamino Accattoli and Delia Kesner. The structural λ-calculus. In 19th EACSL Annual Conference on Computer Science Logic (CSL), Brno, Czech Republic, 2010, volume 6247 of Lecture Notes in Computer Science, pages 381–395. Springer, 2010. doi:10.1007/978-3-642-15205-4_30.

[Bar84] Henk Barendregt. The Lambda Calculus - Its Syntax and Semantics. Elsevier, 1984.

doi:10.1016/c2009-0-14341-6.

[Bar08] Fréric Barral. Decidability for Non-Standard Conversions in Typed Lambda-Calkuli. PhD thesis, Université de Toulouse III - Paul Sabatier and Ludwig-Maximilian Universität München, 2008.

[dC07] Daniel de Carvalho. Sémantiques de la logique linéaire et temps de calcul. PhD thesis, Université de la Méditerranée Aix-Marseille II, 2007.

[ES07] José Espírito Santo. Delayed substitutions. In 18th International Conference on Term Rewriting and Applications (RTA), Paris, France, 2007, volume 4533 of Lecture Notes in Computer Science, pages 169–183. Springer, 2007. doi:10.1007/978-3-540-73449-9_14.

[ES09] José Espírito Santo. The λ-calculus and the unity of structural proof theory. Theory of Computing Systems, 45(4):963–994, 2009. doi:10.1007/s10022-009-9183-9.

[ES16] José Espírito Santo. The polarized λ-calculus. In 11th Workshop on Logical and Semantic Frameworks with Applications (LSFA), Porto, Portugal, 2016, volume 332 of Electronic Notes in Theoretical Computer Science, pages 149–168. Elsevier, 2016. doi:10.1016/j.entcs.2017.04.010.

[ESP03] José Espírito Santo and Luís Pinto. Permutative conversions in intuitionistic multiary sequent calculi with cuts. In 6th International Conference on Typed Lambda Calculi and Applications (TLCA), Valencia, Spain, 2003, volume 2701 of Lecture Notes in Computer Science, pages 286–300. Springer, 2003. doi:10.1007/3-540-44904-3\_20.

[ESP11] José Espírito Santo and Luís Pinto. A calculus of multiary sequent terms. ACM Transactions on Computational Logic, 12(3):22:1–22:41, 2011. doi:10.1145/1929954.1929959.

[Gar94] Philippa Gardner. Discovering needed reductions using type theory. In Masami Hagiya and John C. Mitchell, editors, International Conference on Theoretical Aspects of Computer Software (TACS), Sendai, Japan, 1994, volume 789 of Lecture Notes in Computer Science, pages 555–574. Springer, 1994. doi:10.1007/3-540-57887-0\_115.

[JM00] Felix Joachimski and Ralph Matthes. Standardization and confluence for a lambda calculus with generalized applications. In 11th International Conference on Rewriting Techniques and Applications (RTA), Norwich, UK, 2000, volume 1833 of Lecture Notes in Computer Science, pages 141–155. Springer, 2000. doi:10.1007/10721975_10.
[JM03] Felix Joachimski and Ralph Matthes. Short proofs of normalization for the simply-typed λ-calculus, permutative conversions and gödel’s T. Archive for Mathematical Logic, 42(1), 2003. doi:10.1007/s00153-002-0156-9.

[KP22] Delia Kesner and Loïc Peyrot. Solvability for generalized applications. In 7th International Conference on Formal Structures for Computation and Deduction (FSCD), Haifa, Israel, 2022, volume 228 of Leibniz International Proceedings in Informatics (LIPIcs), pages 18:1–18:22, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.FSCD.2022.18.

[KV15] Delia Kesner and Daniel Ventura. A resource aware computational interpretation for herbelin’s syntax. In 12th International Colloquium on Theoretical Aspects of Computing (ICTAC), Cali, Colombia, 2015, volume 9399 of Lecture Notes in Computer Science, pages 388–403. Springer, 2015. doi:10.1007/978-3-319-25150-9_23.

[KV20] Delia Kesner and Pierre Vial. Non-idempotent types for classical calculi in natural deduction style. Logical Methods in Computer Science; Volume 16, 16(1), 2020. doi:10.23638/LMCS-16(1:3)2020.

[KV22] Delia Kesner and Andrés Viso. Encoding tight typing in a unified framework. In 30th EACSL Annual Conference on Computer Science Logic (CSL), Göttingen, Germany (Virtual Conference), 2022, volume 216 of LIPIcs, pages 27:1–27:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.CSL.2022.27.

[Lev06] Paul Blain Levy. Call-by-push-value: Decomposing call-by-value and call-by-name. Higher-Order and Symbolic Computation, 19(4):377–414, 2006. doi:10.1007/s10990-006-0480-6.

[Mat00] Ralph Matthes. Characterizing strongly normalizing terms for a lambda calculus with generalized applications via intersection types. In Proceedings of the Satellite Workshops of the 27th International Colloquium on Automata, Languages and Programming (ICALP), Geneva, Switzerland, 2000, pages 339-354. Carleton Scientific, Waterloo, Ontario, Canada, 2000.

[Mat01] Ralph Matthes. Interpolation for natural deduction with generalized eliminations. In International Seminar on Proof Theory in Computer Science (PTCS), Dagstuhl Castle, Germany, 2001, volume 2183 of Lecture Notes in Computer Science, pages 153–169. Springer, 2001. doi:10.1007/3-540-45504-3_10.

[Pey22] Loïc Peyrot. From Proof Terms to Programs: An operational and quantitative study of intuitionistic Curry-Howard calculi. PhD thesis, Université Paris Cité, 2022.

[Reg94] Laurent Regnier. Une équivalence sur les lambda-termes. Theoretical Computer Science, 126(2):281–292, 1994. doi:10.1016/0304-3975(94)90012-4.

[SH84a] Peter Schroeder-Heister. Generalized rules for quantifiers and the completeness of the intuitionistic operators &, ∨, ⊃, f, ∀, ∃. In Computation and Proof Theory, pages 399–426. Springer, 1984. doi:10.1007/bfb0099494.

[SH84b] Peter Schroeder-Heister. A natural extension of natural deduction. Journal of Symbolic Logic, 49(4):1284–1300, 1984. doi:10.2307/2274279.

[SH14] Peter Schroeder-Heister. Generalized elimination inferences, higher-level rules, and the implications-as-rules interpretation of the sequent calculus. In Trends in Logic, pages 1–29. Springer Netherlands, 2014. doi:10.1007/978-94-007-7548-0_1.

[Tak95] Masako Takahashi. Parallel reductions in λ-calculus. Information and Computation, 118(1):120–127, 1995. doi:10.1006/inco.1995.1057.

[Ten92] Neil Tennant. Autologic. Edinburgh University Press, Edinburgh, 1992.

[Ten02] Neil Tennant. Ultimate normal forms for parallelized natural deductions. Logic Journal of IGPL, 10(3):299–337, 2002. doi:10.1093/jigpal/10.3.299.

[vP01] Jan von Plato. Natural deduction with general elimination rules. Archive for Mathematical Logic, 40(7):541–567, 2001. doi:10.1007/s001530100091.

[vR96] Femke van Raamsdonk. Confluence and Normalisation for Higher-order Rewriting. PhD thesis, Vrije Universiteit Amsterdam, 1996.