Interpolation for natural deduction with generalized eliminations July 28, 2001

Ralph Matthes

Institut für Informatik der Ludwig-Maximilians-Universität München
Oettingenstraße 67, D-80538 München, Germany
matthes@informatik.uni-muenchen.de

Abstract. A modification of simply-typed λ-calculus by generalized elimination rules in the style of von Plato is presented. Its characteristic feature are permutative conversions also for function types and product types. After the addition of certain extensional reduction rules, an interpolation theorem (à la Lyndon) is proven which is also aware of the terms (a. k. a. the proofs via the Curry-Howard-isomorphism) like in Čubrić's treatment of the usual λ-calculus. Connections between interpolation and canonical liftings of positive and negative type dependencies are given which are important for the intensional treatment of inductive datatypes.

1 Introduction

In logic, interpolation is often used as a tool to reduce arguments on monotone operators to those on positive operators. Mostly, one is satisfied with a logically equivalent positive formula expressing the original monotone dependency. When it comes to functional programming with datatypes, logical equivalence of datatypes is not very helpful. (Think of all the natural numbers being proofs of Nat.) Therefore, a more intensional approach is called for. [Čub94] presented interpolation with additional information: Not only an intermediate formula/type is gained but also proofs/terms for the both implications such that their composition reduces to the original given proof/term in some strongly normalizing and confluent calculus. This idea is taken up in a modification ΛJ of λ-calculus presented in section 2 which has a different application rule than λ-calculus and therefore needs permutative conversions already for function types. Concerning provability/typability, there is no difference with λ-calculus. However, the normal forms have a more appealing structure, made very precise in the last subsection. Section 3 lists several meta-theoretic properties of ΛJ. The interpolation theorem for ΛJ is stated and proven in great detail in section 4. Many subtleties show that in ΛJ one gets an even deeper insight into interpolation than in λ-calculus. Section 5 starts with some material on η-rules and long normal forms, then introduces lifting of arbitrary type dependencies and calculates the interpolants for all of them. In the last subsection, some thoughts are given on the use of interpolation for the purpose of positivization of monotone type dependencies.

Acknowledgement: Many thanks to my former colleague Thorsten Altenkirch who directed my interest towards intensional aspects of the interpolation theorem, and to Jan von Plato who introduced me to the field of generalized eliminations and inspired the present work by his visit to Munich in September 1999. Also thanks to the anonymous referee whose comments were of great help.

2 System ΛJ with Sums and Pairs

We study an extension of simply-typed λ-calculus in the spirit of von Plato [vP98]. The main idea is the use of generalized eliminations, most notably the generalized $\to$-elimination (see below) which is the closure under substitution of the left implication rule of Gentzen, written in the syntax of λ-calculus. For the implicational fragment—called ΛJ—see [JM99]. ΛJ can also be studied in an untyped fashion (hence departing completely from its logical origins): Its confluence and standardization have been established in [JM00]. On the other hand, ΛJ can be assigned intersection types, and thus a characterization of the strongly normalizing terms via typability can be given [Mat00].

2.1 Types and Terms

Types (depending on an infinite supply of type variables denoted α, β—also with decoration) are defined inductively:

$$
\begin{array}{rl}
(\mathrm{V}) & \alpha \\
(\to) & \rho \to \sigma \\
(+) & \rho + \sigma \\
(\times) & \rho \times \sigma \\
(1) & 1 \\
(0) & 0
\end{array}
$$

Let $\mathrm{FV}(\rho)$ be the set of type variables occurring in ρ ("free variables"). We associate $\to$ to the right and abbreviate $\rho_1 \to \ldots \to \rho_n \to \sigma$ by $\boldsymbol{\rho} \to \sigma$.

Typed terms (with types indicated as superscripts or after a colon, and also depending on an infinite supply of variables x^ρ for any type ρ) are defined inductively:

$$
\begin{array}{rl}
(\mathrm{V}) & x^\rho : \rho \\
(\to\text{-I}) & \lambda x^\rho r^\sigma : \rho \to \sigma \\
(\to\text{-E}) & r^{\rho\to\sigma}(s^\rho, x^\sigma.t^\tau) : \tau \\
(+\text{-I}) & \mathrm{inj}_{1,\rho} r^\sigma : \sigma + \rho,\ \mathrm{inj}_{2,\rho} r^\sigma : \rho + \sigma \\
(+\text{-E}) & r^{\rho+\sigma}(x^\rho.s^\tau, y^\sigma.t^\tau) : \tau \\
(\times\text{-I}) & \langle r^\rho, s^\sigma \rangle : \rho \times \sigma \\
(\times\text{-E}) & r^{\rho\times\sigma}(x^\rho.y^\sigma.t^\tau) : \tau \\
(1\text{-I}) & \mathrm{IN1} : 1 \\
(0\text{-E}) & r^0 \rho : \rho
\end{array}
$$

Note that $+$-elimination always [Pra65] has been given in the "generalized style" and that $\times$-elimination is the same as tensor elimination in linear logic[1]. The essential new ingredient (by von Plato) is the generalized $\to$-elimination. In λ-calculus, we would have

$$(\to\text{-E})' \quad r^{\rho\to\sigma} s^{\rho} : \sigma$$
$$(\times\text{-E})' \quad r^{\rho\times\sigma}\mathsf{L} : \rho \text{ and } r^{\rho\times\sigma}\mathsf{R} : \sigma$$

Let $\mathrm{FV}(r)$ be the set of term variables occurring free in r where x is bound in λxr and in $x.t$ (and also for x and y in $x.y.t$ where $x \neq y$ is understood). We identify terms which differ only in the names of their bound variables already on the syntactic level. Hence, e. g., $x(y, z.z)$ and $x(y, y.y)$ are the same term, and α-conversion is kept out of the system. With this identification, it is straightforward to define the result $r^{\rho}[x^{\sigma} := s^{\sigma}]$ of substituting s for x in r (without capture of free variables) as a term of type ρ. In the same way, one can define simultaneous substitution $r^{\rho}[x^{\sigma}, y^{\tau} := s^{\sigma}, t^{\tau}]$ of x by s and y by t (with $x \neq y$) which also gives a term of type ρ.

2.2 Reduction

Rules of β-conversion:

$$(\beta_{\to}) \quad (\lambda x^{\rho} r^{\sigma})(s^{\rho}, y^{\sigma}.t^{\tau}) \rhd t[y := r[x := s]]$$
$$(\beta_{+}) \quad (\mathrm{inj}_{i,\rho_{3-i}} r^{\rho_i})(x_1^{\rho_1}.r_1^{\tau}, x_2^{\rho_2}.r_2^{\tau}) \rhd r_i[x_i := r] \text{ for } i = 1, 2$$
$$(\beta_{\times}) \quad \langle r^{\rho}, s^{\sigma}\rangle(x^{\rho}.y^{\sigma}.t^{\tau}) \rhd t[x, y := r, s]$$

Clearly, if $r \rhd s$ then s has the same type as r ("subject reduction"). Since also $\mathrm{FV}(s) \subseteq \mathrm{FV}(r)$, theses rules are a valid means of transforming proofs (via the Curry-Howard-isomorphism [How80]).

In λ-calculus, we would have

$$(\beta_{\to})' \quad (\lambda x^{\rho} r^{\sigma}) s^{\rho} \rhd r[x := s]$$
$$(\beta_{\times})' \quad \langle r^{\rho}, s^{\sigma}\rangle\mathsf{L} \rhd r \text{ and } \langle r^{\rho}, s^{\sigma}\rangle\mathsf{R} \rhd s$$

It is well-known for λ-calculus with sums that β-reduction alone does not produce terms with the subformula property (every subterm should have a type which forms a part of the type of the term or any of its free variables). In $\Lambda\mathrm{J}$, this already comes from the fragment with function types only, e. g. the term $x^{\alpha\to\alpha}(y^{\alpha}, z^{\alpha}.\lambda u^{\rho\to\rho}u)(\lambda v^{\rho}v, w^{\rho\to\rho}.y) : \alpha$ with free variables $x^{\alpha\to\alpha}$ and y^{α} is β-normal but has a subterm of uncontrolled type ρ.

Let an elimination be anything which may occur as an argument in one of the elimination rules of the term system, hence an object $(s, z.t)$, $(x.s, y.t)$, $(x.y.t)$ or a type (for (0-E)). They will be denoted by $R, S, \ldots$ (For λ-calculus, terms, objects of the form $(x.s, y.t)$, L, R and types would be eliminations.) $\mathrm{FV}(S)$ shall have the obvious meaning as the free variables of S.

Define for any elimination S such that $y^{\tau}S$ is a term, the following eliminations:

[1] For restrictions to linear terms this is much more useful than the two projections, expressed below by L and R.

$(\to\text{-E})$ $(s, x.t^\tau)\{S\} := (s, x.tS)$ where we assume that $x \notin \mathrm{FV}(S)$
$(+\text{-E})$ $(x_1.t_1^\tau, x_2.t_2^\tau)\{S\} := (x_1.t_1S, x_2.t_2S)$ where we assume that $x_1, x_2 \notin \mathrm{FV}(S)$
$(\times\text{-E})$ $(x_1.x_2.t^\tau)\{S\} := (x_1.x_2.tS)$ where we assume that $x_1, x_2 \notin \mathrm{FV}(S)$
(0-E) $\tau\{S\} := \sigma$ with σ the type of the term $x^\tau S$

It is immediate that whenever rRS is a term, then $R\{S\}$ is an elimination and $rR\{S\}$ is also a term.

Rule schema of permutative conversion:

$$(\pi)\quad rRS \rhd rR\{S\}$$

Written out more explicitly, the schema comprises:

$$(\pi_\to)\quad r^{\rho_1\to\rho_2}(s, x.t)S \rhd r(s, x.tS)$$
$$(\pi_+)\quad r^{\rho_1+\rho_2}(x_1.t_1, x_2.t_2)S \rhd r(x_1.t_1S, x_2.t_2S)$$
$$(\pi_\times)\quad r^{\rho_1\times\rho_2}(x_1.x_2.t)S \rhd r(x_1.x_2.tS)$$
$$(\pi_0)\quad (r^0\tau S)^\sigma \rhd r\sigma$$

The rule (π_+) is the usual permutative conversion for sums in natural deduction formulation.

We also introduce the following projection rules (to be justified below):

$$(\kappa_\to)\quad r(s, x.t) \rhd t \text{ for } x \notin \mathrm{FV}(t)$$
$$(\kappa_\times)\quad r(x.y.t) \rhd t \text{ for } x, y \notin \mathrm{FV}(t)$$

Let $\twoheadrightarrow$ be the term closure of all of the above conversions and let $\twoheadrightarrow_0$ be that of all the conversions except the projection rules. Both reduction relations satisfy subject reduction since types are obviously preserved under $\rhd$.

The set NF of normal forms is defined inductively by:

(V) $x \in \mathrm{NF}$.
$(\to\text{-E})$ $s, t \in \mathrm{NF} \wedge y \in \mathrm{FV}(t) \Rightarrow x(s, y.t) \in \mathrm{NF}$.
$(+\text{-E})$ $t_1, t_2 \in \mathrm{NF} \Rightarrow x(x_1.t_1, x_2.t_2) \in \mathrm{NF}$.
$(\times\text{-E})$ $t \in \mathrm{NF} \wedge x_1$ or $x_2 \in \mathrm{FV}(t) \Rightarrow x(x_1.x_2.t) \in \mathrm{NF}$.
(0-E) $x\tau \in \mathrm{NF}$.
$(\to\text{-I})$ $r \in \mathrm{NF} \Rightarrow \lambda xr \in \mathrm{NF}$.
$(+\text{-I})$ $r \in \mathrm{NF} \Rightarrow \mathrm{inj}_i r \in \mathrm{NF}$.
$(\times\text{-I})$ $r_1, r_2 \in \mathrm{NF} \Rightarrow \langle r_1, r_2\rangle \in \mathrm{NF}$.
(1-I) $\mathrm{IN1} \in \mathrm{NF}$.

It can easily be checked that NF is the set of terms which are normal with respect to $\twoheadrightarrow$. Note that unlike λ-calculus, we do not need to consider several arguments to a variable (or even worse for sums and products) but only one elimination.

Define the set NF_0 by removing the requirements on free variables from the definition of NF. Again, it is easy to check that NF_0 characterizes the normal terms with respect to $\twoheadrightarrow_0$.

2.3 Comparison with λ-calculus

Clearly, we can embed λ-calculus into our system by setting $rs := r(s, z.z)$ and $r\mathsf{L} := r(x.y.x)$ and $r\mathsf{R} := r(x.y.y)$ because then $(\lambda xr)s = (\lambda xr)(s, z.z) \to_{\beta_\to} r[x := s]$ and $\langle r, s\rangle\mathsf{L} = \langle r, s\rangle(x.y.x) \to_{\beta_\times} r$ and likewise for $\langle r, s\rangle\mathsf{R}$.

ΛJ can also be encoded in terms of λ-calculus: $r(s, x.t) := t[x := rs]$ and $r(x.y.t) := t[x, y := r\mathsf{L}, r\mathsf{R}]$. This time, only the equalities induced by the rewrite systems are preserved (call $=_\lambda$ that of λ-calculus):

$$\begin{array}{rl}
(\beta_\to) & (\lambda xr)(s, z.t) = t[z := (\lambda xr)s] =_\lambda t[z := r[x := s]] \\
(\beta_\times) & \langle r, s\rangle(x.y.t) = t[x, y := \langle r, s\rangle\mathsf{L}, \langle r, s\rangle\mathsf{R}] =_\lambda t[x, y := r, s] \\
(\pi_\to) & r(s, x.t)S = t[x := rs]S = (tS)[x := rs] = r(s, x.tS) \\
(\pi_\times) & r(x.y.t)S = t[x, y := r\mathsf{L}, r\mathsf{R}]S = (tS)[x, y := r\mathsf{L}, r\mathsf{R}] = r(x.y.tS) \\
(\kappa_\to) & r(s, x.t) = t[x := rs] = t \text{ if } x \notin \mathrm{FV}(t) \\
(\kappa_\times) & r(x.y.t) = t[x, y := r\mathsf{L}, r\mathsf{R}] = t \text{ if } x, y \notin \mathrm{FV}(t)
\end{array}$$

It is well-known that the set of normal λ-terms w. r. t. $(\beta_\to)$', (β_+), $(\beta_\times)$', (π_+) and (π_0)—henceforth the corresponding reduction will be denoted by $\twoheadrightarrow_\lambda$—is inductively characterized by the set NF', defined as follows:

$$\begin{array}{rl}
(\mathrm{V}) & x \in \mathrm{NF}'. \\
(\to\text{-E}) & s, t \in \mathrm{NF}' \wedge y \in \mathrm{FV}(t) \Rightarrow t[y := xs] \in \mathrm{NF}'. \\
(+\text{-E}) & t_1, t_2 \in \mathrm{NF}' \Rightarrow x(x_1.t_1, x_2.t_2) \in \mathrm{NF}'. \\
(\times\text{-E}) & t \in \mathrm{NF}' \wedge x_1 \text{ or } x_2 \in \mathrm{FV}(t) \Rightarrow t[x_1, x_2 := x\mathsf{L}, x\mathsf{R}] \in \mathrm{NF}'. \\
(0\text{-E}) & x\tau \in \mathrm{NF}'. \\
(\to\text{-I}) & r \in \mathrm{NF}' \Rightarrow \lambda xr \in \mathrm{NF}'. \\
(+\text{-I}) & r \in \mathrm{NF}' \Rightarrow \mathrm{inj}_i r \in \mathrm{NF}'. \\
(\times\text{-I}) & r_1, r_2 \in \mathrm{NF}' \Rightarrow \langle r_1, r_2\rangle \in \mathrm{NF}'. \\
(1\text{-I}) & \mathrm{IN1} \in \mathrm{NF}'.
\end{array}$$

Apparently, NF' is nothing but the encoded version of NF. But unlike NF', NF's definition is syntax-directed and does not come as a surprise. Consequently, an analysis of normal forms in λ-calculus can be done more conveniently via a detour through ΛJ. We will see below that a more fine-grained analysis of interpolation can be achieved in this manner.

3 Strong Normalization, Standardization and Confluence

We claim strong normalization of $\twoheadrightarrow$. This would be proved by combining the proofs of strong normalization of ΛJ (with function types only) with that of λ-calculus with sums and permutative conversions in [JM99] since we do not expect any harm from products.

It should be possible to address standardization in a similar way to [JM00] where untyped ΛJ (without case analysis and pairing) has been treated.

We would prefer to give a direct proof of confluence instead of proving local confluence and referring to strong normalization (Newman's Lemma). We believe that again the method in [JM00] should carry over to the full system. (Local confluence can easily be established by considering the critical pairs which are non-trivial, though.)

4 Interpolation

Intuitionistic logic has the interpolation property which means that from a proof of $\rho \to \sigma$ one can find a type τ with type variables among the intersection of those in ρ and σ such that $\rho \to \tau$ and $\tau \to \sigma$ are provable. This would be called Craig interpolation. From the proof, one can see that more can be said about positivity and negativity of the occurrences of the type variables in τ: Positive occurrences in τ imply positive occurrences in both ρ and σ, and similarly for negative occurrences. This is called Lyndon interpolation.

We did not yet exploit our λ-calculus structure in which we formulate intuitionistic reasoning. Since there are terms (a. k. a. proofs) of $r : \rho \to \sigma$, $s : \rho \to \tau$ and $t : \tau \to \sigma$, we may ask whether one can have s and t even such that r and $\lambda x^\rho.t(sx)$ are intensionally equal and not merely proofs of the same statement. [Čub94] established this fact for usual λ-calculus (and extended the result to bicartesian closed categories). In his case, intensional equality for λ-calculus meant equality w. r. t. β-reduction and permutations. Unfortunately, this does not suffice for the treatment of generalized applications. Therefore, we have to extend our notion of reduction by extended permutation rules and then prove Lyndon interpolation in a style quite similar to [Čub94]. However, we will profit a lot from the simpler structure of normal forms in our calculus.

4.1 Adding Some Extensionality

In order to formulate the interpolation theorem we need a more powerful permutation principle for function and product types. The extended permutation rules are:

($\pi^e_\to$) $g[z := r(s, x.t)] \rhd r(s, x.g[z := t])$ for any term g with $z \in \mathrm{FV}(g)$ and (w. l. o. g.) $x \notin \mathrm{FV}(g)$

($\pi^e_\times$) $g[z := r(x.y.t)] \rhd r(x.y.g[z := t])$ for any term g with $z \in \mathrm{FV}(g)$ and (w. l. o. g.) $x, y \notin \mathrm{FV}(g)$

Clearly, the standard permutation rules are the special case with $g := zS$. The extended rules are also justified by our encoding into λ-calculus since in that encoding, we have $g[z := r(s, x.t)] = g[z := t[x := rs]] = r(s, x.g[z := t])$ and $g[z := r(x.y.t)] = g[z := t[x, y := r\mathsf{L}, r\mathsf{R}]] = r(x.y.g[z := t])$.

Let $\twoheadrightarrow_e$ be the reduction relation which is the term closure of the β-rules, (π_+), $(\pi^e_\to)$ and $(\pi^e_\times)$.

The (excluded) degenerate cases with $z \notin \mathrm{FV}(g)$ would read:

($\kappa^r_\to$) $g \rhd r(s, x.g)$ for $x \notin \mathrm{FV}(g)$

($\kappa^r_\times$) $g \rhd r(x.y.g)$ for $x, y \notin \mathrm{FV}(g)$

They are simply the reverses of $(\kappa_\to)$ and $(\kappa_\times)$. Hence, with respect to equality theory, the degenerate cases are already included in $\twoheadrightarrow$.

4.2 The Interpolation Theorem

As a notation, for a set Γ of typed variables with types $\rho_1, \ldots, \rho_n$, write $\Gamma^{\boldsymbol{\rho}}$. Then, $\otimes\boldsymbol{\rho} := \rho_1 \times \ldots \times \rho_n$ (associated to the left). The operator $\otimes$ shall bind stronger than the other connectives.

Let $+(\rho)$ be the set of type variables occurring at some (not necessarily strictly) positive position in ρ and likewise let $-(\rho)$ be the set of type variables occurring negatively in ρ. E.g. $\alpha, \beta \in +((\alpha \to \alpha) \to \beta)$ and only $\alpha \in -((\alpha \to \alpha) \to \beta)$. p (for polarity) will be a variable ranging over $\{+, -\}$. Hence, $\alpha \in p(\rho)$ is a well-defined statement conditional on p's value. Also write $-p$ to mean $-$ in case $p = +$ and $+$ in case $p = -$. More formally, we may now define $+(\rho)$ and $-(\rho)$ as follows:

$$
\begin{array}{rl}
(\mathrm{V}) & +(\alpha) := \{\alpha\},\ -(\alpha) := \emptyset. \\
(\to) & \alpha \in p(\rho \to \sigma) :\Leftrightarrow \alpha \in -p(\rho) \vee \alpha \in p(\sigma) \\
(+) & \alpha \in p(\rho + \sigma) :\Leftrightarrow \alpha \in p(\rho) \vee \alpha \in p(\sigma) \\
(\times) & \alpha \in p(\rho \times \sigma) :\Leftrightarrow \alpha \in p(\rho) \vee \alpha \in p(\sigma) \\
(1) & \alpha \notin p(1) \\
(0) & \alpha \notin p(0)
\end{array}
$$

Theorem 1. Let $r^\rho \in \mathrm{NF}$ and $\mathrm{FV}(r) \subseteq \Pi^{\boldsymbol{\pi}} \uplus \Sigma^{\boldsymbol{\sigma}}$. Set $\Pi_r := \mathrm{FV}(r) \cap \Pi$ and $\Sigma_r := \mathrm{FV}(r) \cap \Sigma$. Then, there is a type $\hat\rho$ and terms $f^{\hat\rho}$ and $\lambda z^{\hat\rho} g^\rho$ both in NF such that

$$
\begin{array}{rl}
(\mathrm{FV}) & \Pi_r \subseteq \mathrm{FV}(f) \subseteq \Pi,\ \Sigma_r \subseteq \mathrm{FV}(\lambda z g) \subseteq \Sigma \text{ and } (z \in \mathrm{FV}(g) \Leftrightarrow \Pi_r \neq \emptyset) \\
& \text{and } z \text{ occurs at most once in } g \\
(+-) & \alpha \in p(\hat\rho) \Rightarrow \alpha \in p(\otimes\boldsymbol{\pi}) \cap p(\boldsymbol{\sigma} \to \rho) \text{ (for all } \alpha \text{ and all values of } p) \\
(\twoheadrightarrow_e) & g[z := f] \twoheadrightarrow_e^* r
\end{array}
$$

Notation: $\mathrm{I}(r, \Pi, \Sigma) = (\hat\rho, f, \lambda z g)$. ($\uplus$ above denotes disjoint union.)

Notice that this theorem is a strengthening of interpolation: By assumption, r proves $\otimes\boldsymbol{\pi} \to \boldsymbol{\sigma} \to \rho$, and a type $\hat\rho$ is produced together with proofs of $\otimes\boldsymbol{\pi} \to \hat\rho$ and $\hat\rho \to \boldsymbol{\sigma} \to \rho$, and every α occurring in $\hat\rho$ also occurs in $\otimes\boldsymbol{\pi}$ and in $\boldsymbol{\sigma} \to \rho$. The interpolation theorem one would like to work with follows with $\boldsymbol{\pi} := \pi$ and $\boldsymbol{\sigma} := \emptyset$.

Also note that from our earlier discussion on NF versus NF$'$, we can infer interpolation for λ-calculus as in [Čub94][2].

Proof. Firstly, the function I is defined by recursion on $r \in \mathrm{NF}$.[3] It is always clear that the second and third component are in NF_0, have the right types and that $\mathrm{FV}(f) \subseteq \Pi$ and $\mathrm{FV}(\lambda z g) \subseteq \Sigma$, and that z occurs at most once in g .

[2] We cannot guarantee that z occurs only once in g, and we will deduce $g[z := f] \twoheadrightarrow_\lambda^* r$ from $g[z := f] =_\lambda r$ and confluence of $\twoheadrightarrow_\lambda$.

[3] We have to be careful with the disjointness assumption for Π and Σ in the recursive calls. Nevertheless, under the renaming convention, we see that in the course of the recursion we never change Π and Σ such that they intersect again.

(triv) Case $\Pi_r = \emptyset$. Set

$$\mathrm{I}(r, \Pi, \Sigma) := (1, \mathrm{IN1}, \lambda z r)$$

with a new variable z. All the other cases are under the proviso "otherwise".
(V) Case x^ρ with $x \in \Pi$. Set

$$\mathrm{I}(x, \Pi, \Sigma) := (\rho, x, \lambda x x).$$

Case x^ρ with $x \in \Sigma$. This is covered by (triv); $\mathrm{I}(x, \Pi, \Sigma) = (1, \mathrm{IN1}, \lambda z x)$.
($\to$-E) Case $x^{\rho_1 \to \rho_2}(s^{\rho_1}, y^{\rho_2}.t^\tau)$ with $x \in \Pi$. We already have

$$\mathrm{I}(s, \Sigma, \Pi) = (\hat{\rho}_1, f_1, \lambda z_1 g_1)$$

(note the interchange of Π and Σ) and

$$\mathrm{I}(t, \Pi \cup \{y\}, \Sigma) = (\hat{\tau}, f_2, \lambda z_2 g_2).$$

Set

$$\mathrm{I}(x(s, y.t), \Pi, \Sigma) := (\hat{\rho}_1 \to \hat{\tau}, \lambda z_1.x(g_1, y.f_2), \lambda z.z(f_1, z_2.g_2)).$$

Case $x^{\rho_1 \to \rho_2}(s^{\rho_1}, y^{\rho_2}.t^\tau)$ with $x \in \Sigma$. We already have

$$\mathrm{I}(s, \Pi, \Sigma) = (\hat{\rho}_1, f_1, \lambda z_1 g_1)$$

and

$$\mathrm{I}(t, \Pi, \Sigma \cup \{y\}) = (\hat{\tau}, f_2, \lambda z_2 g_2).$$

Set

$$\mathrm{I}(x(s, y.t), \Pi, \Sigma) := (\hat{\rho}_1 \times \hat{\tau}, \langle f_1, f_2 \rangle, \lambda z.z(z_1.z_2.x(g_1, y.g_2))).$$

(+-E) Case $x^{\rho_1 + \rho_2}(x_1^{\rho_1}.t_1^\tau, x_2^{\rho_2}.t_2^\tau)$ with $x \in \Pi$. We already have

$$\mathrm{I}(t_i, \Pi \cup \{x_i\}, \Sigma) = (\hat{\tau}_i, f_i, \lambda z_i g_i) \quad \text{for } i = 1, 2.$$

Set $\mathrm{I}(x(x_1.t_1, x_2.t_2), \Pi, \Sigma) :=$

$$(\hat{\tau}_1 + \hat{\tau}_2, x(x_1.\mathrm{inj}_{1,\hat{\tau}_2} f_1, x_2.\mathrm{inj}_{2,\hat{\tau}_1} f_2), \lambda z.z(z_1.g_1, z_2.g_2)).$$

Case $x^{\rho_1 + \rho_2}(x_1^{\rho_1}.t_1^\tau, x_2^{\rho_2}.t_2^\tau)$ with $x \in \Sigma$. We already have

$$\mathrm{I}(t_i, \Pi, \Sigma \cup \{x_i\}) = (\hat{\tau}_i, f_i, \lambda z_i g_i) \quad \text{for } i = 1, 2.$$

Set $\mathrm{I}(x(x_1.t_1, x_2.t_2), \Pi, \Sigma) :=$

$$(\hat{\tau}_1 \times \hat{\tau}_2, \langle f_1, f_2 \rangle, \lambda z.z(z_1.z_2.x(x_1.g_1, x_2.g_2))).$$

($\times$-E) Case $x^{\rho_1 \times \rho_2}(x_1^{\rho_1}.x_2^{\rho_2}.t^\tau)$ with $x \in \Pi$. We already have

$$\mathrm{I}(t, \Pi \cup \{x_1, x_2\}, \Sigma) = (\hat{\tau}, f, \lambda z g).$$

Set

$$\mathrm{I}(x(x_1.x_2.t), \Pi, \Sigma) := (\hat{\tau}, x(x_1.x_2.f), \lambda z g).$$

Case $x^{\rho_1\times\rho_2}(x_1^{\rho_1}.x_2^{\rho_2}.t^\tau)$ with $x\in\Sigma$. We already have

$$\mathrm{I}(t,\Pi,\Sigma\cup\{x_1,x_2\})=(\hat{\tau},f,\lambda zg).$$

Set

$$\mathrm{I}(x(x_1.x_2.t),\Pi,\Sigma):=(\hat{\tau},f,\lambda z.x(x_1.x_2.g)).$$

(0-E) Case $x^0\tau$ with $x\in\Pi$. Set

$$\mathrm{I}(x\tau,\Pi,\Sigma):=(0,x,\lambda x.x\tau).$$

Case $x^0\tau$ with $x\in\Sigma$. (triv) yields $\mathrm{I}(x\tau,\Pi,\Sigma)=(1,\mathrm{IN1},\lambda z.x\tau)$.
($\to$-I) Case $\lambda x^\rho r^\sigma$. We already have $\mathrm{I}(r,\Pi,\Sigma\cup\{x\})=(\hat{\sigma},f,\lambda zg)$. Set

$$\mathrm{I}(\lambda xr,\Pi,\Sigma):=(\hat{\sigma},f,\lambda z\lambda x.g).$$

(+-I) Case $\mathrm{inj}_{i,\rho}r^\sigma$. We already have $\mathrm{I}(r,\Pi,\Sigma)=(\hat{\sigma},f,\lambda zg)$. Set

$$\mathrm{I}(\mathrm{inj}_{i,\rho}r,\Pi,\Sigma):=(\hat{\sigma},f,\lambda z.\mathrm{inj}_{i,\rho}g).$$

($\times$-I) Case $\langle r_1^{\rho_1},r_2^{\rho_2}\rangle$. We already have $\mathrm{I}(r_i,\Pi,\Sigma)=(\hat{\rho}_i,f_i,\lambda z_i g_i)$ for $i=1,2$. Set

$$\mathrm{I}(\langle r_1,r_2\rangle,\Pi,\Sigma):=(\hat{\rho}_1\times\hat{\rho}_2,\langle f_1,f_2\rangle,\lambda z.z(z_1.z_2.\langle g_1,g_2\rangle)).$$

(1-I) Case IN1. Covered by (triv), hence $\mathrm{I}(\mathrm{IN1},\Pi,\Sigma)=(1,\mathrm{IN1},\lambda z.\mathrm{IN1})$.

Because of (triv), we trivially have that $z\in\mathrm{FV}(g)$ implies $\Pi_r\neq\emptyset$.

Concerning the polarity requirement ($+-$) we only deal with an implication elimination $x^{\rho_1\to\rho_2}(s^{\rho_1},y^{\rho_2}.t^\tau)$. The other cases do not need any further sophistication.

Case $x\in\Pi$. Let $\alpha\in p(\hat{\rho}_1\to\hat{\tau})$. Therefore, $\alpha\in -p(\hat{\rho}_1)\cup p(\hat{\tau})$. Show that $\alpha\in p(\otimes\boldsymbol{\pi})\cap p(\boldsymbol{\sigma}\to\tau)$.
  Case $\alpha\in -p(\hat{\rho}_1)$. By induction hypothesis, $\alpha\in -p(\otimes\boldsymbol{\sigma})\cap -p(\boldsymbol{\pi}\to\rho_1)$. Therefore, $\alpha\in p(\boldsymbol{\sigma}\to\tau)$. Also, $\alpha\in p(\otimes\boldsymbol{\pi})\cup -p(\rho_1)$. In case $\alpha\in -p(\rho_1)$, $\alpha\in p(\rho_1\to\rho_2)\subseteq p(\otimes\boldsymbol{\pi})$ because $x^{\rho_1\to\rho_2}\in\Pi$.
  Case $\alpha\in p(\hat{\tau})$. By induction hypothesis, $\alpha\in p(\otimes\boldsymbol{\pi}\times\rho_2)\cap p(\boldsymbol{\sigma}\to\tau)$. Hence, $\alpha\in p(\otimes\boldsymbol{\pi})\cup p(\rho_2)$. If $\alpha\in p(\rho_2)$, then $\alpha\in p(\rho_1\to\rho_2)\subseteq p(\otimes\boldsymbol{\pi})$ again since $x^{\rho_1\to\rho_2}\in\Pi$.
Case $x\in\Sigma$. Let $\alpha\in p(\hat{\rho}_1\times\hat{\tau})$. Therefore, $\alpha\in p(\hat{\rho}_1)\cup p(\hat{\tau})$. Show that $\alpha\in p(\otimes\boldsymbol{\pi})\cap p(\boldsymbol{\sigma}\to\tau)$.
  Case $\alpha\in p(\hat{\rho}_1)$. By induction hypothesis, $\alpha\in p(\otimes\boldsymbol{\pi})\cap p(\boldsymbol{\sigma}\to\rho_1)$. Hence, $\alpha\in p(\boldsymbol{\sigma}\to\tau)\cup p(\rho_1)$. In case $\alpha\in p(\rho_1)$, $\alpha\in -p(\rho_1\to\rho_2)\subseteq p(\boldsymbol{\sigma}\to\tau)$ because $x^{\rho_1\to\rho_2}\in\Sigma$.
  Case $\alpha\in p(\hat{\tau})$. By induction hypothesis, $\alpha\in p(\otimes\boldsymbol{\pi})\cap p(\boldsymbol{\sigma}\to\rho_2\to\tau)$. Hence, $\alpha\in p(\boldsymbol{\sigma}\to\tau)\cup -p(\rho_2)$. If $\alpha\in -p(\rho_2)$, then $\alpha\in -p(\rho_1\to\rho_2)\subseteq p(\boldsymbol{\sigma}\to\tau)$ again since $x^{\rho_1\to\rho_2}\in\Sigma$.

The verification of the statements that f and g have certain free variables, hinges on the tacit assumption in the definition that names of the bound variables are always chosen appropriately (which can be done by the renaming convention). First show that if $z \notin \mathrm{FV}(g)$, then $\Pi_r = \emptyset$: Only the cases (triv), ($\times$-E), ($\to$-I) and (+-I) have to be analyzed. The most interesting case is ($\times$-E)Π: $x \in \Pi$. Therefore, we have to show that $z \in \mathrm{FV}(g)$. This holds by induction hypothesis, since due to $x(x_1.x_2.t) \in \mathrm{NF}$, x_1 or x_2 in $\mathrm{FV}(t)$. The other cases follow directly from the induction hypothesis. Therefore, we have proved

$$\Pi_r \neq \emptyset \Rightarrow z \in \mathrm{FV}(g) \quad (*)$$

Now prove $\Pi_r \subseteq \mathrm{FV}(f)$ and $\Sigma_r \subseteq \mathrm{FV}(\lambda z g)$ together with $f, \lambda z g \in \mathrm{NF}$: (Again we only deal with ($\to$-E) because no further arguments are needed for the other cases.)

Case $x \in \Pi$. $\Pi_{x(s,y.t)} = \{x\} \cup \Pi_s \cup (\Pi_t \setminus \{y\})$. $\Sigma_{x(s,y.t)} = \Sigma_s \cup (\Sigma_t \setminus \{y\})$. $y \in \mathrm{FV}(t)$ (by definition of NF), hence $(\Pi \cup \{y\})_t = \Pi_t \cup \{y\}$. By induction hypothesis, $\Sigma_s \subseteq \mathrm{FV}(f_1)$, $\Pi_s \subseteq \mathrm{FV}(\lambda z_1 g_1)$, $\Pi_t \cup \{y\} \subseteq \mathrm{FV}(f_2)$ and $\Sigma_t \subseteq \mathrm{FV}(\lambda z_2 g_2)$. Therefore, $\Pi_{x(s,y.t)} \subseteq \mathrm{FV}(\lambda z_1.x(g_1, y.f_2))$, $y \in \mathrm{FV}(f_2)$ and $\Sigma_{x(s,y.t)} \subseteq \mathrm{FV}(\lambda z.z(f_1, z_2.g_2))$. Finally, by $(*)$, $z_2 \in \mathrm{FV}(g_2)$.

Case $x \in \Sigma$. $\Pi_{x(s,y.t)} = \Pi_s \cup (\Pi_t \setminus \{y\})$. $\Sigma_{x(s,y.t)} = \{x\} \cup \Sigma_s \cup (\Sigma_t \setminus \{y\})$. $y \in \mathrm{FV}(t)$ (by definition of NF), hence $(\Sigma \cup \{y\})_t = \Sigma_t \cup \{y\}$. We (implicitly) assume that $y \notin \Pi$, hence $\Pi_t \setminus \{y\} = \Pi_t$. By induction hypothesis, $\Pi_s \subseteq \mathrm{FV}(f_1)$, $\Sigma_s \subseteq \mathrm{FV}(\lambda z_1 g_1)$, $\Pi_t \subseteq \mathrm{FV}(f_2)$ and $\Sigma_t \cup \{y\} \subseteq \mathrm{FV}(\lambda z_2 g_2)$. Therefore, $\Pi_{x(s,y.t)} \subseteq \mathrm{FV}(\langle f_1, f_2\rangle)$, $\Sigma_{x(s,y.t)} \subseteq \mathrm{FV}(\lambda z.z(z_1.z_2.x(g_1, y.g_2))$ and $y \in \mathrm{FV}(g_2)$. Because $\Pi_{x(s,y.t)} \neq \emptyset$, $\Pi_s \neq \emptyset$ or $\Pi_t \neq \emptyset$. In the first case, $z_1 \in \mathrm{FV}(g_1)$ by $(*)$, in the second case $z_2 \in \mathrm{FV}(g_2)$ by $(*)$. Hence, by induction hypothesis (and $z_2 \neq y$), $z(z_1.z_2.x(g_1, y.g_2)) \in \mathrm{NF}$.

Finally, we check ($\twoheadrightarrow_e$). Only the reduction is given. The induction hypothesis is not explicitly mentioned. The subcases are simply indicated by Π and Σ.

(triv) $r = r$.

(V) Π: $x = x$.

($\to$-E) Π: $(\lambda z_1.x(g_1, y.f_2))(f_1, z_2.g_2) \twoheadrightarrow_{\beta_\to} g_2[z_2 := x(g_1[z_1 := f_1], y.f_2)] \twoheadrightarrow_e^* g_2[z_2 := x(s, y.f_2)] \twoheadrightarrow_{\pi^e_\to} x(s, y.g_2[z_2 := f_2]) \twoheadrightarrow_e^* x(s, y.t)$. Note that due to $(\Pi \cup \{y\})_t \neq \emptyset$ (by the requirement $y \in \mathrm{FV}(t)$ in the definition of NF), we have $z_2 \in \mathrm{FV}(g_2)$ which allows to apply $(\pi^e_\to)$.
Σ: $\langle f_1, f_2\rangle(z_1.z_2.x(g_1, y.g_2)) \twoheadrightarrow_{\beta_\times} x(g_1[z_1 := f_1], y.g_2[z_2 := f_2]) \twoheadrightarrow_e^* x(s, y.t)$

(+-E) Π: $x(x_1.\mathrm{inj}_{1,\hat{\tau}_2} f_1, x_2.\mathrm{inj}_{2,\hat{\tau}_1} f_2)(z_1.g_1, z_2.g_2) \twoheadrightarrow_{\pi_+}$
$x(x_1.(\mathrm{inj}_{1,\hat{\tau}_2} f_1)(z_1.g_1, z_2.g_2), x_2.(\mathrm{inj}_{2,\hat{\tau}_1} f_2)(z_1.g_1, z_2.g_2)) \twoheadrightarrow_{\beta_+} \twoheadrightarrow_{\beta_+}$
$x(x_1.g_1[z_1 := f_1], x_2.g_2[z_2 := f_2]) \twoheadrightarrow_e^* x(x_1.t_1, x_2.t_2)$
Σ: $\langle f_1, f_2\rangle(z_1.z_2.x(x_1.g_1, x_2.g_2)) \twoheadrightarrow_{\beta_\times} x(x_1.g_1[z_1 := f_1], g_2[z_2 := f_2]) \twoheadrightarrow_e^* x(x_1.t_1, x_2.t_2)$

($\times$-E) Π: $g[z := x(x_1.x_2.f)] \twoheadrightarrow_{\pi^e_\times} x(x_1.x_2.g[z := f]) \twoheadrightarrow_e^* x(x_1.x_2.t)$: Because of $(\Pi \cup \{x_1, x_2\})_t \neq \emptyset$ (by the requirement x_1 or $x_2 \in \mathrm{FV}(t)$ in the definition of NF), we have $z \in \mathrm{FV}(g)$ which allows to apply $(\pi^e_\times)$.
Σ: $x(x_1.x_2.g[z := f]) \twoheadrightarrow_e^* x(x_1.x_2.t)$

(0-E) Π: $x\tau = x\tau$.
($\to$-I) $\lambda x.g[z := f] \twoheadrightarrow_e^* \lambda xr$.
(+-I) $\mathrm{inj}_{i,\rho} g[z := f] \twoheadrightarrow_e^* \mathrm{inj}_{i,\rho} r$.
($\times$-I) $\langle f_1, f_2\rangle(z_1.z_2.\langle g_1, g_2\rangle) \twoheadrightarrow_{\beta_\times} \langle g_1[z_1 := f_1], g_2[z_2 := f_2]\rangle \twoheadrightarrow_e^* \langle r_1, r_2\rangle$ $\square$

Note that $\mathrm{I}(r, \Pi, \Sigma)$ does not depend on variables which are not free in r, i. e. $\mathrm{I}(r, \Pi, \Sigma) = \mathrm{I}(r, \Pi_r, \Sigma_r)$. (However, the case of ($\to$-I) requires the more general definition.)

For the applications we slightly improve the definition of I to a function I' which only differs in the treatment of the case ($\to$-E): In case $x \in \Pi$ and s is a variable $x_1 \in \Pi$, we now set:

$$\mathrm{I}'(x(x_1, y.t), \Pi, \Sigma) := (\hat{\tau}, x(x_1, y.f_2), \lambda z_2 g_2).$$

The justification of the properties (FV), ($+-$) and ($\twoheadrightarrow_e$) is obvious due to $\mathrm{I}(x_1, \Sigma, \Pi) = (1, \mathrm{IN1}, \lambda z_1 x_1)$ and hence

$$\mathrm{I}(x(x_1, y.t), \Pi, \Sigma) = (1 \to \hat{\tau}, \lambda z_1.x(x_1, y.f_2), \lambda z.z(\mathrm{IN1}, z_2.g_2)).$$

In case $x \in \Sigma$ and t is the variable y, we set:

$$\mathrm{I}'(x(s, y.y), \Pi, \Sigma) := (\hat{\rho}_1, f_1, \lambda z_1.x(g_1, y.y)).$$

Again, it is easy to verify (FV), ($+-$) and ($\twoheadrightarrow_e$) by exploiting

$$\mathrm{I}(y, \Pi, \Sigma \cup \{y\}) = (1, \mathrm{IN1}, \lambda z_2 y)$$

and consequently

$$\mathrm{I}(x(s, y.y), \Pi, \Sigma) = (\hat{\rho}_1 \times 1, \langle f_1, \mathrm{IN1}\rangle, \lambda z.z(z_1.z_2.x(g_1, y.y)).$$

Let us illustrate that extended permutation rules were indeed needed.

Example 1. Set $r := x^{\rho\to\sigma}(u^\rho, y^\sigma.\lambda v^\tau y^\sigma)$. By (V), $I(u, \{u\}, \{x\}) = (\rho, u, \lambda uu)$ and $I(y, \{x, y\}, \{u, v\}) = (\sigma, y, \lambda yy)$. By ($\to$-I), $I(\lambda vy, \{x, y\}, \{u\}) = (\sigma, y, \lambda y\lambda vy)$. Hence $I(r, \{x\}, \{u\}) = (\rho \to \sigma, \lambda u.x(u, y.y), \lambda z.z(u, y.\lambda vy))$. But we only have $(z(u, y.\lambda vy))[z := \lambda u.x(u, y.y)] \twoheadrightarrow_{\beta_\to} (\lambda vy)[y := x(u, y.y)] = \lambda v.x(u, y.y)$ and cannot reach r by help of $\twoheadrightarrow^*$.

5 Some Uses of the Interpolation Algorithm

In the first subsection, additional concepts are given in preparation for the statements on the interpolation of liftings.

5.1 Long Normal Forms and Eta-Reduction

Define the set LNF of long normal forms by keeping all the rules of the definition of NF (with NF replaced by LNF) except (V) which now reads

(V) If $x : \alpha$ or $x : 0$ or $x : 1$, then $x \in \mathrm{LNF}$.

Lemma 1. The interpolation functions I and I′ preserve long normal forms, i.e., for $r^\rho \in \mathrm{LNF}$, $\mathrm{I}(r, \Pi, \Sigma) = (\hat{\rho}, f, \lambda zg)$ fulfills $f, \lambda zg \in \mathrm{LNF}$, and similarly for I′.

Proof. Immediate by induction on $r \in \mathrm{LNF}$. □

Consider the following conversions:

$(\eta_\to)$ $\lambda x^\rho.r^{\rho\to\sigma}(x, y^\sigma.y) \rhd r$, if $x \notin \mathrm{FV}(r)$
(η_+) $r^{\rho+\sigma}(x^\rho.\mathrm{inj}_{1,\sigma}x, y^\sigma.\mathrm{inj}_{2,\rho}y) \rhd r$
$(\eta_\times)$ $r^{\rho\times\sigma}(x^\rho.y^\sigma.\langle x, y\rangle) \rhd r$

Let $\twoheadrightarrow_\eta$ be the term closure of those three rules. The η-rules for λ-calculus would have the following differences:

$(\eta_\to)$' $\lambda x^\rho.r^{\rho\to\sigma}x \rhd r$, if $x \notin \mathrm{FV}(r)$
$(\eta_\times)$' $\langle r\mathsf{L}, r\mathsf{R}\rangle \rhd r$

Clearly, the above η-rules of ΛJ are justifiable in our encoding of ΛJ into λ-calculus since, in that encoding, $\lambda x.r(x, y.y) = \lambda x.rx$ and $r(x.y.\langle x, y\rangle) = \langle r\mathsf{L}, r\mathsf{R}\rangle$.

Define η-expansion $\eta(x^\rho) \in \mathrm{LNF}$ for variables by recursion on ρ as follows:

(triv) If ρ is a type variable or $\rho \in \{0, 1\}$, then $\eta(x) := x$.
$(\to)$ $\eta(x^{\rho\to\sigma}) := \lambda y^\rho.x(\eta(y), z^\sigma.\eta(z))$.
$(+)$ $\eta(x^{\rho+\sigma}) := x(y^\rho.\mathrm{inj}_{1,\sigma}\eta(y), z^\sigma.\mathrm{inj}_{2,\rho}\eta(z))$.
$(\times)$ $\eta(x^{\rho\times\sigma}) := x(y^\rho.z^\sigma.\langle\eta(y), \eta(z)\rangle)$.

Notice that λ-calculus does not allow such an easy η-expansion of variables.

5.2 Lifting of Positive and Negative Type Dependencies

It is well-known that if a type variable α occurs only positively in some type ρ, one can "lift" terms of type $\sigma \to \tau$ to terms of type $\rho[\alpha := \sigma] \to \rho[\alpha := \tau]$ (see e.g. [Lei90]). We pursue a new approach which does not use any positivity requirement for the definition (but will give the usual definition in case it is met).

But first, we have to introduce type substitution: $\rho[\alpha := \sigma]$ shall denote the result of replacing every occurrence of α in ρ by σ. Similarly, define the result $\rho[\alpha, \beta := \sigma, \tau]$ of simultaneous substitution of α by σ and β by τ in ρ (we assume $\alpha \neq \beta$). Type substitution and simultaneous type substitution may also be carried out for terms, denoted by $r[\alpha := \sigma]$ and $r[\alpha, \beta := \sigma, \tau]$, respectively.

Associate with every type variable α a "fresh" type variable α^-. Let α^p be α for $p = +$ and let α^{-p} be α^- for $p = +$ and α for $p = -$. Set likewise for term variables h and h_-, $h_p := h$ for $p = +$ and $h_{-p} := h_-$ for $p = +$ and $h_{-p} := h$ for $p = -$.

Define for every type ρ and every type variable α a term $\mathrm{lift}^p_{\lambda\alpha\rho}$ of type ρ with free variables among the typed variables $h : \alpha^{-p} \to \alpha^p$, $h_- : \alpha^p \to \alpha^{-p}$ and $x : \rho[\alpha := \alpha^-]$ by recursion on ρ (write $r[\alpha \leftrightarrow \alpha^-]$ for $r[\alpha, \alpha^- := \alpha^-, \alpha]$):

(triv) If ρ is a type variable $\neq \alpha$ or $\rho \in \{0, 1\}$, set $\mathrm{lift}^p_{\lambda\alpha\rho} := x$ (which is type-correct!)

(V) $\mathrm{lift}^p_{\lambda\alpha\alpha} := h_p(x, y^\alpha.y)$. (Note that $h_p : \alpha^- \to \alpha$.)

($\to$) $\mathrm{lift}^p_{\lambda\alpha.\rho_1\to\rho_2} :=$

$$\lambda x^{\rho_1}.x^{(\rho_1\to\rho_2)[\alpha:=\alpha^-]}(\mathrm{lift}^{-p}_{\lambda\alpha\rho_1}[\alpha \leftrightarrow \alpha^-], x^{\rho_2[\alpha:=\alpha^-]}.\mathrm{lift}^p_{\lambda\alpha\rho_2}).$$

Note that $\mathrm{lift}^{-p}_{\lambda\alpha\rho_1}[\alpha \leftrightarrow \alpha^-]$ has free variables among $h : \alpha^{-p} \to \alpha^p$, $h^- : \alpha^p \to \alpha^{-p}$ and x^{ρ_1}.

($+$) $\mathrm{lift}^p_{\lambda\alpha.\rho_1+\rho_2} :=$

$$x^{(\rho_1+\rho_2)[\alpha:=\alpha^-]}(x^{\rho_1[\alpha:=\alpha^-]}.\mathrm{inj}_{1,\rho_2}\mathrm{lift}^p_{\lambda\alpha\rho_1}, x^{\rho_2[\alpha:=\alpha^-]}.\mathrm{inj}_{2,\rho_1}\mathrm{lift}^p_{\lambda\alpha\rho_2}).$$

($\times$) $\mathrm{lift}^p_{\lambda\alpha.\rho_1\times\rho_2} :=$

$$x^{(\rho_1\times\rho_2)[\alpha:=\alpha^-]}(x^{\rho_1[\alpha:=\alpha^-]}.x^{\rho_2[\alpha:=\alpha^-]}.\langle\mathrm{lift}^p_{\lambda\alpha\rho_1}, \mathrm{lift}^p_{\lambda\alpha\rho_2}\rangle).$$

It should be clear that x with differently written types means different variables (even if ρ_1 and ρ_2 happen to be the same type!).

Lemma 2. (i) $\mathrm{lift}^p_{\lambda\alpha\rho} \in \mathrm{LNF}$ and $x^{\rho[\alpha:=\alpha^-]} \in \mathrm{FV}(\mathrm{lift}^p_{\lambda\alpha\rho})$.
(ii) $\alpha \in p(\rho) \Leftrightarrow h \in \mathrm{FV}(\mathrm{lift}^p_{\lambda\alpha\rho})$ and $\alpha \in -p(\rho) \Leftrightarrow h_- \in \mathrm{FV}(\mathrm{lift}^p_{\lambda\alpha\rho})$.

Proof. By induction on ρ. □

By a slight extension of our notation with a variable q ranging over $\{+, -\}$, we may rewrite (ii) to $\alpha \in qp(\rho) \Leftrightarrow h_q \in \mathrm{FV}(\mathrm{lift}^p_{\lambda\alpha\rho})$.

For applications, we normally do not want that h_- occurs in $\mathrm{lift}^p_{\lambda\alpha\rho}$. Therefore, we define the set $\mathrm{Only}_+(\rho)$ of type variables occurring only positively in ρ and the set $\mathrm{Only}_-(\rho)$ of type variables occurring only negatively in ρ by setting $\alpha \in \mathrm{Only}_p(\rho) :\Leftrightarrow \alpha \notin -p(\rho)$. It is clear that we get the following recursive (with respect to ρ) characterization of $\mathrm{Only}_+(\rho)$ and $\mathrm{Only}_-(\rho)$:

(V) $\mathrm{Only}_+(\alpha) =$ all variables, $\mathrm{Only}_-(\alpha) =$ all variables except α.
($\to$) $\mathrm{Only}_p(\rho \to \sigma) = \mathrm{Only}_{-p}(\rho) \cap \mathrm{Only}_p(\sigma)$.
($+$) $\mathrm{Only}_p(\rho + \sigma) = \mathrm{Only}_p(\rho) \cap \mathrm{Only}_p(\sigma)$.
($\times$) $\mathrm{Only}_p(\rho \times \sigma) = \mathrm{Only}_p(\rho) \cap \mathrm{Only}_p(\sigma)$.
(1) $\mathrm{Only}_p(1) =$ all variables.
(0) $\mathrm{Only}_p(0) =$ all variables.

Corollary 1. If $\alpha \in \mathrm{Only}_p(\rho)$, then $h_- \notin \mathrm{FV}(\mathrm{lift}^p_{\lambda\alpha\rho})$.

Lemma 3. If $\alpha \notin \mathrm{FV}(\rho)$, then $\mathrm{lift}^p_{\lambda\alpha\rho} = \eta(x)$.

Proof. Induction on ρ. □

5.3 Interpolation and Lifting

Lemma 4. (i) $\mathrm{I}'(\mathrm{lift}^p_{\lambda\alpha\rho}, \{h, h_-, x\}, \emptyset) = (\rho, \mathrm{lift}^p_{\lambda\alpha\rho}, \lambda z.\eta(z))$.
(ii) $\mathrm{I}'(\mathrm{lift}^p_{\lambda\alpha\rho}, \{x\}, \{h, h_-\}) = (\rho[\alpha := \alpha^-], \eta(x), \lambda x.\mathrm{lift}^p_{\lambda\alpha\rho})$.

Proof. By induction on ρ. Note that we never use the case (triv) of the interpolation theorem. The case (V) needs the modifications made for I'. Only ($\to$) (i) will be dealt with: By induction hypothesis (i) and (ii)

$$\mathrm{I}'(\mathrm{lift}^p_{\lambda\alpha\rho_2}, \{h, h_-, x^{\rho_2[\alpha:=\alpha^-]}\}, \emptyset) = (\rho_2, \mathrm{lift}^p_{\lambda\alpha\rho_2}, \lambda z_2.\eta(z_2)) \quad \text{and}$$

$$\mathrm{I}'(\mathrm{lift}^{-p}_{\lambda\alpha\rho_1}, \{x^{\rho_1[\alpha:=\alpha^-]}\}, \{h^{\alpha^p \to \alpha^{-p}}, h_-^{\alpha^{-p} \to \alpha^p}\}) =$$

$$(\rho_1[\alpha := \alpha^-], \eta(x^{\rho_1[\alpha:=\alpha^-]}), \lambda x^{\rho_1[\alpha:=\alpha^-]}.\mathrm{lift}^{-p}_{\lambda\alpha\rho_1}).$$

Therefore, $\mathrm{I}'(\mathrm{lift}^{-p}_{\lambda\alpha\rho_1}[\alpha \leftrightarrow \alpha^-], \{x^{\rho_1}\}, \{h^{\alpha^{-p} \to \alpha^p}, h_-^{\alpha^p \to \alpha^{-p}}\}) =$

$$(\rho_1, \eta(x^{\rho_1}), \lambda x^{\rho_1}.\mathrm{lift}^{-p}_{\lambda\alpha\rho_1}[\alpha \leftrightarrow \alpha^-]).$$

Finally (recall that we may drop superfluous variables from Π and Σ),

$$\mathrm{I}'(\mathrm{lift}^p_{\lambda\alpha.\rho_1 \to \rho_2}, \{h, h_-, x^{(\rho_1 \to \rho_2)[\alpha:=\alpha^-]}\}, \emptyset) =$$

$$(\rho_1 \to \rho_2, \mathrm{lift}^p_{\lambda\alpha.\rho_1 \to \rho_2}, \lambda z \lambda x^{\rho_1}.z(\eta(x^{\rho_1}), z_2.\eta(z_2))).$$

□

For a fixed type variable α we now define ρ^+ which is ρ with every negative occurrence of α replaced by α^- (recall that α^- shall be a "fresh" type variable) and ρ^- which is $\rho^+[\alpha \leftrightarrow \alpha^-]$. Then $\alpha \in \mathrm{Only}_p(\rho^p)$ and $\alpha^- \in \mathrm{Only}_{-p}(\rho^p)$, in short: $\alpha^q \in \mathrm{Only}_{qp}(\rho^p)$. More formally, we define ρ^p for fixed α:

- (triv) If ρ is a type variable $\neq \alpha$ or $\rho \in \{0, 1\}$, then set $\rho^p := \rho$.
- (V) α^p is defined as before.
- ($\to$) $(\rho \to \sigma)^p := \rho^{-p} \to \sigma^p$.
- ($+$) $(\rho + \sigma)^p := \rho^p + \sigma^p$.
- ($\times$) $(\rho \times \sigma)^p := \rho^p \times \sigma^p$.

It is clear that $\mathrm{FV}(\rho^p) \subseteq \mathrm{FV}(\rho) \cup \{\alpha^-\}$, that $\rho^{-p} = \rho^p[\alpha \leftrightarrow \alpha^-]$, $\rho^p[\alpha^- := \alpha] = \rho$ and $\alpha^q \in \mathrm{Only}_{qp}(\rho^p)$. [Concerning the last statement, consider its equivalent $\alpha^q \notin -qp(\rho^p)$. (V): If $p = q$, then $\alpha^p \notin -(\alpha^p)$. If $p \neq q$, then $\alpha^{-p} \notin +(\alpha^p)$. ($\to$): Show $\alpha^q \notin -qp(\rho^{-p} \to \sigma^p)$, i.e., $\alpha \notin -q(-p)(\rho^{-p})$ and $\alpha \notin -qp(\sigma^p)$ which both follow from the induction hypothesis.]

Lemma 5. Let $x : \rho[\alpha := \alpha^-]$, $h : \alpha^{-p} \to \alpha^p$, $h_- : \alpha^p \to \alpha^{-p}$ (hence $h_q : \alpha^{-qp} \to \alpha^{qp}$, and therefore $h_p : \alpha^- \to \alpha$) and let $\hat{\alpha}$ be a "fresh" type variable.

Then $\mathrm{I}'(\mathrm{lift}^p_{\lambda\alpha\rho}, \{x, h_q\}, \{h_{-q}\}) =$

$$(\rho^{qp}, \mathrm{lift}^p_{\lambda\alpha.\rho^{qp}[\alpha^-:=\hat{\alpha}]}[\hat{\alpha} := \alpha^-], \lambda x^{\rho^{qp}}.\mathrm{lift}^p_{\lambda\alpha.\rho^{-qp}[\alpha^-:=\hat{\alpha}]}[\hat{\alpha} := \alpha]).$$

Proof. Induction on ρ (quite tedious). $\square$

The preceding two lemmas may be written down in a uniform notation as follows:

$$\mathrm{I}'(\mathrm{lift}^p_{\lambda\alpha\rho}, \{x\} \cup \Pi, \Sigma) =$$
$$(\rho^*, \mathrm{lift}^p_{\lambda\alpha.\rho^*[\alpha^- := \hat{\alpha}]}[\hat{\alpha} := \alpha^-], \lambda x^{\rho^*}.\mathrm{lift}^p_{\lambda\alpha.\rho^*[\alpha := \hat{\alpha}][\alpha^- := \alpha]}[\hat{\alpha} := \alpha])$$

$$\text{with } \rho^* := \begin{cases} \rho & \text{, if } h, h_- \in \Pi \\ \rho[\alpha := \alpha^-] & \text{, if } h, h_- \in \Sigma \\ \rho^p & \text{, if } h \in \Pi,\ h_- \in \Sigma \\ \rho^{-p} & \text{, if } h_- \in \Pi,\ h \in \Sigma \end{cases}$$

5.4 Application: Positivization of Monotone Types

Consider a type ρ and a type variable α such that there is a term $m : \rho$ with free variables among $h : \alpha^- \to \alpha$ and $x : \rho[\alpha := \alpha^-]$. We know that if $\alpha \in \mathrm{Only}_+(\rho)$, then $\mathrm{lift}^+_{\lambda\alpha\rho}$ is such a term. By lemma 4, interpolation (the function I') gives back the type ρ and that term together with an η-expansion of the identity. What do we get for arbitrary $m \in \mathrm{NF}$? We get

$$\mathrm{I}'(m, \{h, x\}, \emptyset) = (\hat{\rho}, f^{\hat{\rho}}, \lambda z^{\hat{\rho}} g^{\rho})$$

with $f, \lambda zg \in \mathrm{NF}$, $\mathrm{FV}(f) \subseteq \{h, x\}$, $\mathrm{FV}(\lambda zg) = \emptyset$, $\mathrm{FV}(\hat{\rho}) \subseteq \mathrm{FV}(\rho)$ (especially, $\alpha^- \notin \mathrm{FV}(\hat{\rho})$), $\alpha \in \mathrm{Only}_+(\hat{\rho})$ and $g[z := f] \twoheadrightarrow^*_e m$. A proof for the last but one property: If α were in $-(\hat{\rho})$, then $\alpha \in -((\alpha^- \to \alpha) \times \rho[\alpha := \alpha^-])$ which is not true. Moreover,

$$\lambda x^\rho.f[\alpha^- := \alpha][h^{\alpha\to\alpha} := \lambda y^\alpha y]$$

is a closed term of type $\rho \to \hat{\rho}$. Therefore, we found for $\lambda\alpha\rho$ and m a positivization: A type $\hat{\rho}$ with $\alpha \in \mathrm{Only}_+(\rho)$ such that there are closed terms of types $\hat{\rho} \to \rho$ and $\rho \to \hat{\rho}$. We even have that

$$g[z := f[\alpha^- := \alpha][h^{\alpha\to\alpha} := \lambda y^\alpha y]] = g[z := f][\alpha^- := \alpha][h^{\alpha\to\alpha} := \lambda y^\alpha y] \twoheadrightarrow^*_e$$
$$m[\alpha^- := \alpha][h^{\alpha\to\alpha} := \lambda y^\alpha y].$$

Therefore, if $m[\alpha^- := \alpha][h^{\alpha\to\alpha} := \lambda y^\alpha y] \twoheadrightarrow^*_e x$ holds, we have even a retract from $\hat{\rho}$ to ρ. But this cannot be expected without adding the η-conversions: It is easy to see that

$$\mathrm{lift}^p_{\lambda\alpha\rho}[\alpha^- := \alpha][h^{\alpha\to\alpha}, h^{\alpha\to\alpha}_- := \lambda y^\alpha y, \lambda y^\alpha y] \twoheadrightarrow^*_\eta x.$$

Therefore, we could hope for a retract with respect to $\twoheadrightarrow_e \cup \twoheadrightarrow_\eta$. But this seems to hold only in the trivial case of $\alpha \in \mathrm{Only}_+(\rho)$ where we do not need any positivization.

I conjecture that something like

$$\mathrm{lift}^+_{\lambda\alpha\hat{\rho}}[x^{\hat{\rho}[\alpha:=\alpha^-]} := f[\alpha := \alpha^-][h^{\alpha^-\to\alpha^-} := \lambda y^{\alpha^-} y]] \twoheadrightarrow^*_e f$$

holds. In the case of $\alpha \in \mathrm{Only}_+(\rho)$, it can easily be shown by Lemma 4 (without the help of η-reduction): In fact one has to show (for arbitrary ρ and α):

Lemma 6. Two easy properties of liftings:
(i) $\mathrm{lift}^p_{\lambda\alpha\rho}[x^{\rho[\alpha:=\alpha^-]} := \mathrm{lift}^p_{\lambda\alpha\rho}[\alpha := \alpha^-][h^{\alpha^-\to\alpha^-}, h_-^{\alpha^-\to\alpha^-} := \lambda y^{\alpha^-}y, \lambda y^{\alpha^-}y] \twoheadrightarrow^*_e$ $\mathrm{lift}^p_{\lambda\alpha\rho}$.
(ii) $\mathrm{lift}^p_{\lambda\alpha\rho}[\alpha^- := \alpha][h^{\alpha\to\alpha}, h_-^{\alpha\to\alpha} := \lambda y^\alpha y, \lambda y^\alpha y][x^\rho := \mathrm{lift}^p_{\lambda\alpha\rho}] \twoheadrightarrow^*_e \mathrm{lift}^p_{\lambda\alpha\rho}$.

Proof. Routine verification by simultaneous induction on ρ. (Note that $(\pi^e_\times)$ is not needed beyond $(\pi_\times)$.) □

Let us finally study an interesting example of positivization.

Example 2. Let $\rho := (((\alpha \to \beta) \to \alpha) \to \alpha) \to \alpha$. Set

$$m^\rho := \lambda y^{((\alpha\to\beta)\to\alpha)\to\alpha}.y\Big(\lambda z^{\alpha\to\beta}.x\Big(\lambda z_1^{(\alpha^-\to\beta)\to\alpha^-}.z_1 S, u_4^{\alpha^-}.h(u_4, u_5^\alpha.u_5)\Big), u_6^\alpha.u_6\Big)$$

with

$$S := (\lambda z_2^{\alpha^-}.h(z_2, u_1^\alpha.z(u_1, u_2^\beta.u_2)), u_3^{\alpha^-}.u_3)$$

and the only free variables $h : \alpha^- \to \alpha$ and $x : \rho[\alpha := \alpha^-]$. It is an easy exercise to calculate $I'(r, \{x, h\}, \emptyset) = ((\alpha \to \beta) \to \alpha, \ldots)$. The shown type may be seen as the optimal positivization. Thorsten Altenkirch found another one, namely $(((\alpha \to \beta) \to \alpha) \to \beta) \to \alpha$, which can also be found by using a more complicated term m. Since we are not dealing with uniform interpolation [Pit92], the interpolating types may depend on the given monotonicity proof m.

6 Conclusions

In the light of the definitional embedding of the system ΛJ with generalized eliminations into usual λ-calculus, e. g. via $r(s, z.t) \mapsto t[z := rs]$, we arrived at a more fine-grained analysis of Lyndon interpolation even when taking into account the proofs/terms and not only provability/typability. Moreover, the proof of the results on the system with generalized eliminations shows how elegant reasoning becomes by induction on the normal forms obtainable in the system in contrast to those in the standard formulation of λ-calculus resp. derivations in intuitionistic logic.

Finally, a new formulation of liftings has been presented whose behaviour under interpolation could have hardly been studied in standard λ-calculus due to notational burdens already in the definition of the result of interpolation.

The following questions certainly have to be addressed in the future: What is the behaviour of the extended permutation rules and of η-reduction? Is it possible to operationalize η-expansion and prove the usual theorems about it? Is it possible to prove strong normalization of our system by continuation-passing-style transformations in the spirit of [dG99]? Is there a nice categorical semantics of ΛJ which would allow to infer our theorem from the general result in [Čub94]? And, of course, the conjecture on the previous page should be settled since it would allow to reduce iteration and primitive recursion on monotone inductive types to those on positive inductive types without recourse to impredicativity (compare [Mat01]).

References

[Čub94] Djordje Čubrić. Interpolation property for bicartesian closed categories. *Archive for Mathematical Logic*, 33:291–319, 1994.

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

[How80] W. A. Howard. The formulae-as-types notion of construction. In J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 479–490. Academic Press, 1980.

[JM99] Felix Joachimski and Ralph Matthes. Short proofs of normalization for the simply-typed lambda-calculus, permutative conversions and Gödel's T. *Archive for Mathematical Logic*, 1999. Accepted for publication.

[JM00] Felix Joachimski and Ralph Matthes. Standardization and confluence for a lambda calculus with generalized applications. In Leo Bachmair, editor, *Rewriting Techniques and Applications, Proceedings of the 11th International Conference RTA 2000, Norwich, UK*, volume 1833 of *Lecture Notes in Computer Science*, pages 141–155. Springer Verlag, 2000.

[Lei90] Daniel Leivant. Contracting proofs to programs. In Piergiorgio Odifreddi, editor, *Logic and Computer Science*, volume 31 of *APIC Studies in Data Processing*, pages 279–327. Academic Press, 1990.

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

[Mat01] Ralph Matthes. Tarski's fixed-point theorem and lambda calculi with monotone inductive types. To appear in Synthese, 2001.

[Pit92] Andrew Pitts. On an interpretation of second order quantification in first order intuitionistic propositional logic. *The Journal of Symbolic Logic*, 57(1):33–52, 1992.

[Pra65] Dag Prawitz. *Natural Deduction. A Proof-Theoretical Study.* Almquist and Wiksell, 1965.

[vP98] Jan von Plato. Natural deduction with general elimination rules. Submitted, 1998.