Call-by-Name Reduction and Cut-Elimination in Classical Logic

Kentaro Kikuchi

RIEC, Tohoku University
Katahira 2-1-1, Aoba-ku, Sendai 980-8577, Japan
kentaro@nue.riec.tohoku.ac.jp

Abstract. We present a version of Herbelin's $\overline{\lambda}\mu$-calculus in the call-by-name setting to study the precise correspondence between normalization and cut-elimination in classical logic. Our translation of $\lambda\mu$-terms into a set of terms in the calculus does not involve any administrative redexes, in particular η-expansion on μ-abstraction. The isomorphism preserves β, μ-reduction, which is simulated by a local-step cut-elimination procedure in the typed case, where the reduction system strictly follows the "cut=redex" paradigm. We show that the underlying untyped calculus is confluent and enjoys the PSN (preservation of strong normalization) property for the isomorphic image of $\lambda\mu$-calculus, which in turn yields a confluent and strongly normalizing local-step cut-elimination procedure for classical logic.

1 Introduction

The Curry-Howard correspondence between proofs and programs is traditionally explained through natural deduction systems. In recent years, it has been recognized that the correspondence is also suitable for sequent calculus style systems in a significant way. A key system there was developed by Herbelin [9, 10], who introduced a sequent calculus in which a unique cut-free proof is associated to each normal term of the simply typed λ-calculus. He provided a term notation for proofs of the sequent calculus and defined a reduction system for those terms where reduction steps correspond to cut-elimination steps. The close connection to the simply typed λ-calculus makes the reduction system, called $\overline{\lambda}$-calculus, a clue to understand computational contents of cut-elimination. In fact, Herbelin already observed in [9, 10] that a reduction rule corresponding to permutation of cuts is necessary to simulate full β-reduction, and that his calculus does not have such a reduction rule.

Curien and Herbelin's $\overline{\lambda}\mu\tilde{\mu}$-calculus [2] is a classical variant of the $\overline{\lambda}$-calculus, which exhibits symmetries such as term/context and call-by-name/call-by-value. Although this system shows many good aspects of considering sequent calculus as a typing system, it lacks some points that the original $\overline{\lambda}$-calculus has. First, as remarked in Section 2 of [2], the $\overline{\lambda}\mu\tilde{\mu}$-calculus does not entirely follow the "cut=redex" paradigm: there exist cuts that are not redexes. Secondly, it is not a conservative extension of Parigot's $\lambda\mu$-calculus [15]: it cannot distinguish the

terms translated from two different $\lambda\mu$-terms $\mu\alpha.[\alpha](MN)$ and MN (see also [18] where it is shown that translations between $\lambda\mu$-terms and $\overline{\lambda}\mu\tilde{\mu}$-terms are bijective only modulo linear reductions). As a consequence, the $\overline{\lambda}\mu\tilde{\mu}$-calculus involves μ-reduction in simulating β-reduction. Since μ-abstraction corresponds to the (RAA) rule in classical natural deduction, this means that one cannot separate the intuitionistic part from the whole classical system.

On the other hand, it is known that desired results can be obtained if restricted to the intuitionistic case. Espírito Santo [7] and Dyckhoff and Urban [6] pointed out that there is a precise bijection between λ-terms and a subset of $\overline{\lambda}$-terms. Then they added (global [7] or local [6]) reduction rules to the $\overline{\lambda}$-calculus so that the resulting systems can simulate full β-reduction. While Espírito Santo proved the confluence and strong normalization of his calculus relying on those of proof nets [8] through a technique by Danos et al. [4], Dyckhoff and Urban obtained more general results using Bloo and Geuvers' method [1] for explicit substitution calculi. This was possible because the process of local-step cut-elimination has a strong similarity to the propagation of explicit substitutions that correspond to the cut rules.

The purpose of this paper is to extend Dyckhoff and Urban's work to the case of classical logic and obtain a local-step cut-elimination procedure that simulates normalization in classical natural deduction in the strict sense. The sequent calculus we use is essentially the same as the one introduced by Herbelin [10], who also defined reduction rules for the proof terms, which were not enough to simulate β, μ-reduction. We modify some of his reduction rules and add some reduction rules that are necessary to simulate full β, μ-reduction. Although the addition of the reduction rules introduces many critical pairs, we can prove the confluence of our calculus, even in the untyped case. Moreover, the untyped calculus enjoys the PSN (preservation of strong normalization) property with respect to β, μ-reduction, which at once allows us to prove strong normalization for typed terms, and hence, for the local-step cut-elimination procedure.

One of the features of Herbelin-style systems is the use of a distinguished position, called *stoup*, in each side of a sequent. Among several systems based on classical sequent calculus with stoup [3, 10, 2, 20], the system we adopt differs from the others in the use of sequents where the stoups on both left and right sides are filled with formulas. This leads to the syntactic category of lists of terms (without involving parameterization by continuation variables), which allows us to obtain the precise bijection between $\lambda\mu$-terms and a subset of terms in our calculus. We define reduction relations on the subset of terms that exactly correspond to β-reduction and μ-reduction. To do this, we need four meta-operations. Our calculus can be considered in some sense a calculus making those meta-operations explicit in the same way as usual explicit substitution calculi make the usual meta-substitution explicit.

As regards related work, Urban [19] studied a local-step cut-elimination procedure for classical sequent calculus without stoup. He proved strong normalization of the cut-elimination procedure and derived from it strong normalization of the simply typed λ-calculus, but it was not related to reductions in the $\lambda\mu$-

Table 1. $\overline{\lambda}\mu$x-terms and typing rules

$$t, u, v ::= xl \mid \lambda x.t \mid \mu\alpha.[\gamma]t \mid tl \mid t\langle v/x\rangle \mid t\langle {}^{[\alpha](-l)}/_{[\alpha]-}\rangle$$
$$l, l' ::= [] \mid t :: l \mid ll' \mid l\langle v/x\rangle \mid l\langle {}^{[\alpha](-l')}/_{[\alpha]-}\rangle$$

$$\frac{}{\Gamma; A \vdash [] : A; \Delta}\ Ax \qquad \frac{\Gamma, x : A; A \vdash l : B; \Delta}{\Gamma, x : A; - \vdash xl : B; \Delta}\ Der$$

$$\frac{\Gamma; - \vdash t : A; \Delta \quad \Gamma; B \vdash l : C; \Delta}{\Gamma; A \supset B \vdash t :: l : C; \Delta}\ L\supset \qquad \frac{\Gamma, x : A; - \vdash t : B; \Delta}{\Gamma; - \vdash \lambda x.t : A \supset B; \Delta}\ R\supset$$

$$\frac{\Gamma; - \vdash t : A; \alpha : B, \Delta}{\Gamma; - \vdash \mu\alpha.[\gamma]t : B; \Delta}\ Chg \qquad \frac{\Gamma; \Pi \vdash a : A; \Delta \quad \Gamma; A \vdash l : B; \Delta}{\Gamma; \Pi \vdash al : B; \Delta}\ Cut_1$$

where $\alpha \not\equiv \gamma$ implies $\gamma : A \in \Delta$

$$\frac{\Gamma; - \vdash v : A; \Delta \quad \Gamma, x : A; \Pi \vdash a : B; \Delta}{\Gamma; \Pi \vdash a\langle v/x\rangle : B; \Delta}\ Cut_2$$

$$\frac{\Gamma; \Pi \vdash a : B; \alpha : A, \Delta \quad \Gamma; A \vdash l : C; \Delta}{\Gamma; \Pi \vdash a\langle {}^{[\alpha](-l)}/_{[\alpha]-}\rangle : B; \alpha : C, \Delta}\ Cut_\mu$$

calculus. The relation between systems with and without stoup was suggested in Section 4 of [2], and formally discussed in [13]. In general, a system without stoup can be considered as a special case of systems with stoup where neither of left and right stoups is filled with a formula. Strong normalization for the $\overline{\lambda}\mu\tilde{\mu}$-calculus with explicit substitutions was investigated in [11, 17].

The present paper is organized as follows. In Section 2 we introduce the calculus and typing system. In Section 3 we consider the subset of terms that correspond to usual $\lambda\mu$-terms. In Section 4 we study a subcalculus that plays an important role in our proofs. In Section 5 we explain how to simulate β, μ-reduction in the calculus, and prove confluence. In Section 6 we prove PSN and strong normalization for typed terms. In Section 7 we briefly discuss some variations of our calculus.

To save space we omit some of the proofs, but a full version with all proofs is available at `http://www.nue.riec.tohoku.ac.jp/user/kentaro/clc.pdf`.

2 The $\overline{\lambda}\mu$x-Calculus

Table 1 presents the syntax and typing rules of the $\overline{\lambda}\mu$x-calculus. The syntax has two kinds of expressions: terms and lists of terms, ranged over by t, u, v and by

l, l', respectively. The set of terms is denoted by $\mathcal{T}_{\overline{\lambda}\mu\mathtt{x}}$ and the set of lists of terms by $\mathcal{L}_{\overline{\lambda}\mu\mathtt{x}}$. Elements of $\mathcal{T}_{\overline{\lambda}\mu\mathtt{x}} \cup \mathcal{L}_{\overline{\lambda}\mu\mathtt{x}}$ are called $\overline{\lambda}\mu\mathtt{x}$-*terms* and ranged over by a, b. The notions of free and bound variables are defined as usual, with an additional clause that the variable x in $a\langle v/x\rangle$ binds the free occurrences of x in a (similarly for α in $a\langle {}^{[\alpha](-l)}/_{[\alpha]-}\rangle$). The set of free variables of a $\overline{\lambda}\mu\mathtt{x}$-term a is denoted by $FV(a)$. The symbol $\equiv$ denotes syntactic equality modulo α-conversion.

The typing system is a sequent calculus based on two kinds of judgements: $\Gamma; - \vdash t : B; \Delta$ and $\Gamma; A \vdash l : B; \Delta$. We use $\Gamma; \Pi \vdash a : B; \Delta$ to denote both kinds of judgements, with Π being zero or one formula. Since a list of terms l can be considered as a context with a hole in the position of head variable, we can consistently use, in the typing rule Cut_1, the notation al, which is read as the $\overline{\lambda}\mu\mathtt{x}$-term obtained by filling the hole of l with a term or another context a.

The notion of $\overline{\lambda}\mu\mathtt{x}$-reduction is defined by the contextual closures of all reduction rules in Table 2. We use $\rightarrow_{\overline{\lambda}\mu\mathtt{x}}$ for one-step reduction, $\overset{+}{\rightarrow}_{\overline{\lambda}\mu\mathtt{x}}$ for its transitive closure, and $\overset{*}{\rightarrow}_{\overline{\lambda}\mu\mathtt{x}}$ for its reflexive transitive closure. The set of $\overline{\lambda}\mu\mathtt{x}$-terms that are strongly normalizing with respect to $\overline{\lambda}\mu\mathtt{x}$-reduction is denoted by $\mathcal{SN}_{\overline{\lambda}\mu\mathtt{x}}$. These kinds of notations are also used for the notions of other reductions introduced in this paper.

The subcalculus of $\overline{\lambda}\mu\mathtt{x}$ without $(Beta)$, (Mu_1) and (Mu_2) is denoted by $\mathtt{x}$. This subcalculus plays an important role in this paper and is studied in detail in Section 4.

Herbelin's original $\overline{\lambda}\mu$-calculus [10] has the same typing rules as $\overline{\lambda}\mu\mathtt{x}$, but has different reduction rules, which are not enough to simulate β, μ-reduction of the $\lambda\mu$-calculus. In particular, the rules (App_6), $(Subst_7)$ and $(Musubst_7)$, which are necessary for proving Lemma 1, are absent from Herbelin's original calculus.

The reduction rules of $\overline{\lambda}\mu\mathtt{x}$-calculus also define a cut-elimination procedure for typing derivations of $\overline{\lambda}\mu\mathtt{x}$-terms, which ensures that this typing system has the subject reduction property. In Appendix A, we display the cut-elimination steps corresponding to $\overline{\lambda}\mu\mathtt{x}$-reduction for typed terms. Notice that the cut-elimination steps corresponding to (Mu_1) and (Mu_2) are not very natural, since to simulate one-step μ-reduction one needs to decompose the argument list (a similar observation is found in Section 2 of [2]). A natural cut-elimination step for Mu_1-redex is shown in Section 7, but it does not simulate one-step μ-reduction.

3 Pure Terms

Table 3 presents the syntax of *pure terms*, which are the subset of $\overline{\lambda}\mu\mathtt{x}$-terms that correspond to terms (and lists of terms) of usual $\lambda\mu$-calculus. (The syntax and typing rules of the simply typed $\lambda\mu$-calculus are found in Appendix B.) The grammar of pure terms, which has the explicit construction of lists of terms, is close to the following inductive characterization of the set of $\lambda\mu$-terms:

$$\begin{aligned} M, N ::= {} & xM_1 \dots M_n \mid \lambda x.M \mid (\lambda x.M)NM_1 \dots M_n \\ & \mid \mu\alpha.[\gamma]M \mid (\mu\alpha.[\gamma]M)NM_1 \dots M_n \quad (n \geqslant 0). \end{aligned}$$

Table 2. Reduction rules of $\overline{\lambda}\mu\tilde{\mu}$-calculus

$$
\begin{array}{lrl}
(Beta) & (\lambda x.t)(u :: l) & \rightarrow t\langle u/x\rangle l \\
(Mu_1) & (\mu\alpha.[\gamma]t)(u :: l) & \rightarrow (\mu\alpha.[\gamma]t\langle {}^{[\alpha](-(u::[]))}/_{[\alpha]-}\rangle)l \qquad (\alpha \not\equiv \gamma) \\
(Mu_2) & (\mu\alpha.[\alpha]t)(u :: l) & \rightarrow (\mu\alpha.[\alpha]t\langle {}^{[\alpha](-(u::[]))}/_{[\alpha]-}\rangle(u :: []))l \\
(App_1) & []l & \rightarrow l \\
(App_2) & (u :: l)l' & \rightarrow u :: (ll') \\
(App_3) & (xl)l' & \rightarrow x(ll') \\
(App_4) & (\lambda x.t)[] & \rightarrow \lambda x.t \\
(App_5) & (\mu\alpha.[\gamma]t)[] & \rightarrow \mu\alpha.[\gamma]t \\
(App_6) & (al)l' & \rightarrow a(ll') \\
(Subst_1) & []\langle v/x\rangle & \rightarrow [] \\
(Subst_2) & (u :: l)\langle v/x\rangle & \rightarrow u\langle v/x\rangle :: l\langle v/x\rangle \\
(Subst_3) & (yl)\langle v/x\rangle & \rightarrow yl\langle v/x\rangle \qquad (y \not\equiv x) \\
(Subst_4) & (xl)\langle v/x\rangle & \rightarrow vl\langle v/x\rangle \\
(Subst_5) & (\lambda y.t)\langle v/x\rangle & \rightarrow \lambda y.t\langle v/x\rangle \\
(Subst_6) & (\mu\alpha.[\gamma]t)\langle v/x\rangle & \rightarrow \mu\alpha.[\gamma]t\langle v/x\rangle \\
(Subst_7) & (al)\langle v/x\rangle & \rightarrow a\langle v/x\rangle l\langle v/x\rangle \\
(Musubst_1) & []\langle {}^{[\alpha](-l')}/_{[\alpha]-}\rangle & \rightarrow [] \\
(Musubst_2) & (u :: l)\langle {}^{[\alpha](-l')}/_{[\alpha]-}\rangle & \rightarrow u\langle {}^{[\alpha](-l')}/_{[\alpha]-}\rangle :: l\langle {}^{[\alpha](-l')}/_{[\alpha]-}\rangle \\
(Musubst_3) & (xl)\langle {}^{[\alpha](-l')}/_{[\alpha]-}\rangle & \rightarrow xl\langle {}^{[\alpha](-l')}/_{[\alpha]-}\rangle \\
(Musubst_4) & (\lambda x.t)\langle {}^{[\alpha](-l')}/_{[\alpha]-}\rangle & \rightarrow \lambda x.t\langle {}^{[\alpha](-l')}/_{[\alpha]-}\rangle \\
(Musubst_5) & (\mu\delta.[\gamma]t)\langle {}^{[\alpha](-l')}/_{[\alpha]-}\rangle & \rightarrow \mu\delta.[\gamma]t\langle {}^{[\alpha](-l')}/_{[\alpha]-}\rangle \qquad (\gamma \not\equiv \alpha) \\
(Musubst_6) & (\mu\delta.[\alpha]t)\langle {}^{[\alpha](-l')}/_{[\alpha]-}\rangle & \rightarrow \mu\delta.[\alpha](t\langle {}^{[\alpha](-l')}/_{[\alpha]-}\rangle l') \\
(Musubst_7) & (al)\langle {}^{[\alpha](-l')}/_{[\alpha]-}\rangle & \rightarrow a\langle {}^{[\alpha](-l')}/_{[\alpha]-}\rangle l\langle {}^{[\alpha](-l')}/_{[\alpha]-}\rangle
\end{array}
$$

Table 3. Pure terms

$$t, u, v ::= xl \mid \lambda x.t \mid (\lambda x.t)(u :: l) \mid \mu\alpha.[\gamma]t \mid (\mu\alpha.[\gamma]t)(u :: l)$$
$$l, l' ::= [] \mid t :: l$$

$$(\beta) \quad (\lambda x.t)(u :: l) \rightarrow \{t[u/x]\}l$$
$$(\mu) \quad (\mu\alpha.[\gamma]t)(u :: l) \rightarrow \{\mu\alpha.([\gamma]t)^{[[\alpha]\{-\}(u::[])}/_{[\alpha]-}]\}l$$

where the meta-operations $\_/\_]$ and $\{\_\}\_$ are defined as follows:

$$\begin{aligned}
(yl)[v/x] &=_{def} yl[v/x] \qquad (y \not\equiv x)\\
(xl)[v/x] &=_{def} \{v\}l[v/x]\\
(\lambda y.t)[v/x] &=_{def} \lambda y.t[v/x]\\
((\lambda y.t)(u :: l))[v/x] &=_{def} (\lambda y.t[v/x])(u[v/x] :: l[v/x])\\
(\mu\alpha.[\gamma]t)[v/x] &=_{def} \mu\alpha.[\gamma]t[v/x]\\
((\mu\alpha.[\gamma]t)(u :: l))[v/x] &=_{def} (\mu\alpha.[\gamma]t[v/x])(u[v/x] :: l[v/x])\\
[][v/x] &=_{def} []\\
(u :: l)[v/x] &=_{def} u[v/x] :: l[v/x]\\
\{xl\}l' &=_{def} x(l@l')\\
\{\lambda y.t\}[] &=_{def} \lambda y.t\\
\{\lambda y.t\}(u :: l) &=_{def} (\lambda y.t)(u :: l)\\
\{(\lambda y.t)(u :: l)\}l' &=_{def} (\lambda y.t)(u :: (l@l'))\\
\{\mu\alpha.[\gamma]t\}[] &=_{def} \mu\alpha.[\gamma]t\\
\{\mu\alpha.[\gamma]t\}(u :: l) &=_{def} (\mu\alpha.[\gamma]t)(u :: l)\\
\{(\mu\alpha.[\gamma]t)(u :: l)\}l' &=_{def} (\mu\alpha.[\gamma]t)(u :: (l@l'))\\
[]@l &=_{def} l\\
(u :: l)@l' &=_{def} u :: (l@l')
\end{aligned}$$

Table 4. Translations Ψ and Θ

$$
\begin{array}{rl}
\Psi(x) & =_{def}\ x[] \\
\Psi(MN) & =_{def}\ \{\Psi(M)\}(\Psi(N) :: []) \\
\Psi(\lambda x.M) & =_{def}\ \lambda x.\Psi(M) \\
\Psi(\mu\alpha.[\gamma]M) & =_{def}\ \mu\alpha.[\gamma]\Psi(M)
\end{array}
\qquad
\begin{array}{rl}
\Theta(xl) & =_{def}\ \Theta'(x,l) \\
\Theta(\lambda x.t) & =_{def}\ \lambda x.\Theta(t) \\
\Theta((\lambda x.t)(u :: l)) & =_{def}\ \Theta'(\lambda x.\Theta(t), u :: l) \\
\Theta(\mu\alpha.[\gamma]t) & =_{def}\ \mu\alpha.[\gamma]\Theta(t) \\
\Theta((\mu\alpha.[\gamma]t)(u :: l)) & =_{def}\ \Theta'(\mu\alpha.[\gamma]\Theta(t), u :: l) \\
\Theta'(M,[]) & =_{def}\ M \\
\Theta'(M, u :: l) & =_{def}\ \Theta'(M\Theta(u), l)
\end{array}
$$

For the definition of β-reduction on pure terms, we need two meta-operations $[\_/\_]$ and $\{\_\}\_$. The operation $[\_/\_]$ is basically meta-substitution, but the result of substitution is not in general a pure term (e.g., the $\overline{\lambda}\mu\mathtt{x}$-term $(x[])[]$, which is obtained by substituting $x[]$ for y in $y[]$, is not a pure term). So we use the operation $\{\_\}\_$ (and also $\_@\_$) to append the first list to the residual list in such cases. The operation $[^{[\alpha]\{-\}(u::[])}/_{[\alpha]-}]$ in the μ-rule replaces inductively each occurrence in $[\gamma]t$ of the form $[\alpha]v$ by $[\alpha]\{v\}(u :: [])$. (A formal definition of the operation $[^{[\alpha]\{-\}l}/_{[\alpha]-}]$ is omitted.)

The operation $\{\_\}\_$ was introduced in [6] for the intuitionistic case, called *generalized application* (a similar operation is found in [7]). In the $\overline{\lambda}\mu\tilde{\mu}$-calculus [2], the process of the operation is implemented with the help of (linear) μ-reductions. We distinguish the process from usual μ-reduction by directly mapping the two operands of generalized application to the corresponding pure term.

Now translations between pure terms and $\lambda\mu$-terms are given in Table 4.

Proposition 1. *$\Theta \circ \Psi = id$ and $\Psi \circ \Theta = id$.*

Theorem 1. *For any $\lambda\mu$-terms M, M',*

1. *$M \to_\beta M'$ if and only if $\Psi(M) \to_\beta \Psi(M')$,*
2. *$M \to_\mu M'$ if and only if $\Psi(M) \to_\mu \Psi(M')$.*

Later we show that the translations preserve the types of terms and that β, μ-reduction on pure terms can be simulated by $\overline{\lambda}\mu\mathtt{x}$-reduction, which reveals how to simulate normalization in classical natural deduction by cut-elimination in Herbelin's sequent calculus.

4 Properties of the Subcalculus x

In this section we study properties of the subcalculus x which is obtained from $\overline{\lambda}\mu\mathtt{x}$-calculus by deleting the $(Beta)$, (Mu_1) and (Mu_2) rules. In the typed case, it corresponds to the cut-elimination steps except the key-cases, i.e., the cases

where the cut-formula is introduced into the stoups of the conclusions of both left and right subderivations over the cut rule.

First we show that the normal forms of the subcalculus $\mathtt{x}$ coincide with pure terms.

Proposition 2. *Let $a \in \mathcal{T}_{\overline{\lambda}\mu\mathtt{x}} \cup \mathcal{L}_{\overline{\lambda}\mu\mathtt{x}}$. a is a pure term if and only if a is in $\mathtt{x}$-normal form.*

Proof. The only if part is by induction on the structure of pure terms. We prove the if part by induction on the structure of a. Suppose that a is in $\mathtt{x}$-normal form. Then by the induction hypothesis, all subterms of a are pure. Now, if a is not pure, then a is one of the forms $bl (\not\equiv (\lambda x.t)(u :: l')$ or $(\mu\alpha.[\gamma]t)(u :: l'))$, $b\langle v/x\rangle$ and $b\langle {}^{[\alpha](-l)}/_{[\alpha]-}\rangle$ where b, l, v are pure. In any case we see that a is an $\mathtt{x}$-redex, which is a contradiction. □

Next we show that the subcalculus $\mathtt{x}$ is strongly normalizing.

Proposition 3. *The subcalculus $\mathtt{x}$ is strongly normalizing.*

Proof. The proof is by interpretation, extending the one in Appendix A of [5]. We define a function $h : \mathcal{T}_{\overline{\lambda}\mu\mathtt{x}} \cup \mathcal{L}_{\overline{\lambda}\mu\mathtt{x}} \longrightarrow \mathbb{N}$ as follows:

$$
\begin{array}{ll}
h([]) & =_{def} 1 \\
h(u :: l) & =_{def} h(u) + h(l) + 1 \\
h(xl) & =_{def} h(l) + 1 \\
h(\lambda x.t) & =_{def} h(t) + 1 \\
h(\mu\alpha.[\gamma]t) & =_{def} h(t) + 1 \\
h(al) & =_{def} 2 \times h(a) + h(l) + 1 \\
h(a\langle v/x\rangle) & =_{def} (3 \times h(v) + 1) \times h(a) \\
h(a\langle {}^{[\alpha](-l)}/_{[\alpha]-}\rangle) & =_{def} (3 \times h(l) + 1)^{h(a)}
\end{array}
$$

and observe that if $a \to_{\mathtt{x}} b$ then $h(a) > h(b)$. Another proof uses the lexicographic path ordering induced by a first-order encoding found in Section 6, without annotation of natural numbers. □

Since the subcalculus $\mathtt{x}$ has many critical pairs, we prove its confluence using a projection of $\overline{\lambda}\mu\mathtt{x}$-terms into pure terms rather than Newman's Lemma. For this we prove the following important lemma, which shows that the subcalculus $\mathtt{x}$ correctly simulates the meta-operations on pure terms.

Lemma 1. *Let t, v, l, l', a be pure terms with $t, v \in \mathcal{T}_{\overline{\lambda}\mu\mathtt{x}}$, $l, l' \in \mathcal{L}_{\overline{\lambda}\mu\mathtt{x}}$ and $a \in \mathcal{T}_{\overline{\lambda}\mu\mathtt{x}} \cup \mathcal{L}_{\overline{\lambda}\mu\mathtt{x}}$. Then*

1. $ll' \overset{*}{\to}_{\mathtt{x}} l@l'$,
2. $tl \overset{*}{\to}_{\mathtt{x}} \{t\}l$,
3. $a\langle v/y\rangle \overset{*}{\to}_{\mathtt{x}} a[v/y]$,
4. $a\langle {}^{[\alpha](-l)}/_{[\alpha]-}\rangle \overset{*}{\to}_{\mathtt{x}} a[{}^{[\alpha]\{-\}l}/_{[\alpha]-}]$.

Proof. The first part is by induction on the structure of l. The second part is by a case analysis according to the form of tl. The remaining two parts are proved by induction on the structure of a. □

The above lemma may be regarded as a weak normalization result for the subcalculus $\mathtt{x}$, since we may use it to reduce any innermost $\mathtt{x}$-redex repeatedly until all such redexes are eliminated. We define $\mathtt{x}(a)$ as the $\mathtt{x}$-normal form of a obtained in this way.

Definition 1. *For each $a \in \mathcal{T}_{\overline{\lambda\mu}\mathtt{x}} \cup \mathcal{L}_{\overline{\lambda\mu}\mathtt{x}}$, the (pure) term $\mathtt{x}(a)$ is defined inductively as follows:*

1. $\mathtt{x}([]) =_{def} []$,
2. $\mathtt{x}(t :: l) =_{def} \mathtt{x}(t) :: \mathtt{x}(l)$,
3. $\mathtt{x}(ll') =_{def} \mathtt{x}(l)@\mathtt{x}(l')$,
4. $\mathtt{x}(yl) =_{def} y\mathtt{x}(l)$,
5. $\mathtt{x}(\lambda y.t) =_{def} \lambda y.\mathtt{x}(t)$,
6. $\mathtt{x}(\mu\alpha.[\gamma]t) =_{def} \mu\alpha.[\gamma]\mathtt{x}(t)$,
7. $\mathtt{x}(tl) =_{def} \{\mathtt{x}(t)\}\mathtt{x}(l)$,
8. $\mathtt{x}(a\langle v/y\rangle) =_{def} \mathtt{x}(a)[\mathtt{x}(v)/y]$,
9. $\mathtt{x}(a\langle {}^{[\alpha](-l)}/_{[\alpha]-}\rangle) =_{def} \mathtt{x}(a)[{}^{[\alpha]\{-\}\mathtt{x}(l)}/_{[\alpha]-}]$.

Remark 1. For any pure term $a \in \mathcal{T}_{\overline{\lambda\mu}\mathtt{x}} \cup \mathcal{L}_{\overline{\lambda\mu}\mathtt{x}}$, $\mathtt{x}(a) \equiv a$.

Lemma 2. *Let $a \in \mathcal{T}_{\overline{\lambda\mu}\mathtt{x}} \cup \mathcal{L}_{\overline{\lambda\mu}\mathtt{x}}$. Then $a \xrightarrow{*}_{\mathtt{x}} \mathtt{x}(a)$.*

Proof. By induction on the structure of a, using Lemma 1. □

Lemma 3. *Let $a, b \in \mathcal{T}_{\overline{\lambda\mu}\mathtt{x}} \cup \mathcal{L}_{\overline{\lambda\mu}\mathtt{x}}$. If $a \to_{\mathtt{x}} b$, then $\mathtt{x}(a) \equiv \mathtt{x}(b)$.*

Proof. By induction on the structure of a. If the $\mathtt{x}$-reduction is not at the root then the lemma easily follows from the induction hypothesis. If the $\mathtt{x}$-reduction is at the root, e.g., if $a \equiv (\mu\delta.[\alpha]t)\langle {}^{[\alpha](-l)}/_{[\alpha]-}\rangle \to_{\mathtt{x}} \mu\delta.[\alpha](t\langle {}^{[\alpha](-l)}/_{[\alpha]-}\rangle l) \equiv b$ then

$$\begin{aligned}
\mathtt{x}((\mu\delta.[\alpha]t)\langle {}^{[\alpha](-l)}/_{[\alpha]-}\rangle) &\equiv \mathtt{x}(\mu\delta.[\alpha]t)[{}^{[\alpha]\{-\}\mathtt{x}(l)}/_{[\alpha]-}] \\
&\equiv (\mu\delta.[\alpha]\mathtt{x}(t))[{}^{[\alpha]\{-\}\mathtt{x}(l)}/_{[\alpha]-}] \\
&\equiv \mu\delta.[\alpha]\{\mathtt{x}(t)[{}^{[\alpha]\{-\}\mathtt{x}(l)}/_{[\alpha]-}]\}\mathtt{x}(l) \\
&\equiv \mu\delta.[\alpha]\{\mathtt{x}(t\langle {}^{[\alpha](-l)}/_{[\alpha]-}\rangle)\}\mathtt{x}(l) \\
&\equiv \mu\delta.[\alpha]\mathtt{x}(t\langle {}^{[\alpha](-l)}/_{[\alpha]-}\rangle l) \\
&\equiv \mathtt{x}(\mu\delta.[\alpha](t\langle {}^{[\alpha](-l)}/_{[\alpha]-}\rangle l)).
\end{aligned}$$

Some of the other cases need properties of meta-operations on pure terms. □

An immediate consequence of these lemmas is the confluence of the subcalculus $\mathtt{x}$.

Proposition 4. *The subcalculus $\mathtt{x}$ is confluent.*

Proof. Suppose that $a \xrightarrow{*}_{\mathtt{x}} a_1$ and $a \xrightarrow{*}_{\mathtt{x}} a_2$. Then by Lemma 3, $\mathtt{x}(a) \equiv \mathtt{x}(a_i)$ $(i = 1, 2)$. Since $a_i \xrightarrow{*}_{\mathtt{x}} \mathtt{x}(a_i)$ by Lemma 2, $\mathtt{x}(a)$ is a common reduct of a_1 and a_2. □

5 Simulation of β, μ-Reduction

In this section we study the relation between $\rightarrow_{\beta,\mu}$ and $\rightarrow_{\overline{\lambda}\mu\mathtt{x}}$, which in the typed case correspond to normalization and (local-step) cut-elimination, respectively. It is important that on the one hand, $\overline{\lambda}\mu\mathtt{x}$-reduction simulates β, μ-reduction, and on the other hand, $\overline{\lambda}\mu\mathtt{x}$-reduction is sound in regard to β, μ-reduction (i.e., a pure term a is $\overline{\lambda}\mu\mathtt{x}$-reducible to another pure term b only if a is β, μ-reducible to b).

First we show that $\overline{\lambda}\mu\mathtt{x}$-reduction simulates β, μ-reduction.

Theorem 2. *For any pure terms $a, b \in \mathcal{T}_{\overline{\lambda}\mu\mathtt{x}} \cup \mathcal{L}_{\overline{\lambda}\mu\mathtt{x}}$, if $a \rightarrow_{\beta,\mu} b$ then $a \stackrel{+}{\rightarrow}_{\overline{\lambda}\mu\mathtt{x}} b$.*

Proof. By induction on the structure of a. We treat the case $a \equiv (\mu\alpha.[\gamma]t)(u :: l)$ $(\alpha \not\equiv \gamma)$ and $b \equiv \{\mu\alpha.([\gamma]t)[^{[\alpha]\{-\}(u::[])}/_{[\alpha]-}]\}l (\equiv \{\mu\alpha.[\gamma]t[^{[\alpha]\{-\}(u::[])}/_{[\alpha]-}]\}l)$. Then use $\rightarrow_{Mu_1}$ to create $(\mu\alpha.[\gamma]t\langle^{[\alpha](-(u::[]))}/_{[\alpha]-}\rangle)l$, and use Lemma 1 (4) and (2) to reach $\{\mu\alpha.[\gamma]t[^{[\alpha]\{-\}(u::[])}/_{[\alpha]-}]\}l$. □

The strictness in Theorem 2 has a nice consequence.

Corollary 1. *Let $a \in \mathcal{T}_{\overline{\lambda}\mu\mathtt{x}} \cup \mathcal{L}_{\overline{\lambda}\mu\mathtt{x}}$. If $a \in \mathcal{SN}_{\overline{\lambda}\mu\mathtt{x}}$ then $\mathtt{x}(a) \in \mathcal{SN}_{\beta,\mu}$.*

Proof. Suppose that $\mathtt{x}(a) \notin \mathcal{SN}_{\beta,\mu}$. Using Theorem 2 we get an infinite $\overline{\lambda}\mu\mathtt{x}$-reduction sequence starting with $\mathtt{x}(a)$. Since $a \stackrel{*}{\rightarrow}_{\overline{\lambda}\mu\mathtt{x}} \mathtt{x}(a)$ by Lemma 2, we have $a \notin \mathcal{SN}_{\overline{\lambda}\mu\mathtt{x}}$. □

Next we consider the typed case to show the relation between normalization in classical natural deduction and cut-elimination in Herbelin's sequent calculus. For this we first show that the translations in Section 3 preserve the types of terms, defining translations on typing derivations.

Proposition 5. *For any $\lambda\mu$-term M, $\Gamma \vdash M : A; \Delta$ if and only if $\Gamma; - \vdash \Psi(M) : A; \Delta$.*

Proof. The only if part is by induction on the derivation of $\Gamma \vdash M : A; \Delta$. For the if part, it suffices to prove the following by simultaneous induction on the derivations for pure terms:

1. if $\Gamma; - \vdash t : A; \Delta$ then $\Gamma \vdash \Theta(t) : A; \Delta$,
2. if $\Gamma; A \vdash l : B; \Delta$ then $\Gamma \vdash \Theta'(M, l) : B; \Delta$ for any M with $\Gamma \vdash M : A; \Delta$. □

Now we see that the proof of Theorem 2 indicates how to simulate normalization in classical natural deduction by cut-elimination in Herbelin's sequent calculus. Specifically, a redex in natural deduction is translated into one of the key-cases corresponding to, say, a Mu_1-redex $(\mu\alpha.[\gamma]t)(u :: l)$. Then transformation is performed as in Appendix A to create the proof corresponding to $(\mu\alpha.[\gamma]t\langle^{[\alpha](-(u::[]))}/_{[\alpha]-}\rangle)l$, followed by cut-elimination steps to reach the proof corresponding to $\{\mu\alpha.[\gamma]t[^{[\alpha]\{-\}(u::[])}/_{[\alpha]-}]\}l$. The latter cut-elimination steps are in fact strongly normalizing and confluent, since they correspond to reduction steps of the subcalculus $\mathtt{x}$.

Next we show that the *Beta*-reduction and Mu_1, Mu_2-reduction project onto β-reduction and μ-reduction, respectively.

Lemma 4 (Projection). *Let* $a, b \in \mathcal{T}_{\overline{\lambda}\mu\mathtt{x}} \cup \mathcal{L}_{\overline{\lambda}\mu\mathtt{x}}$.

1. *If* $a \to_{Beta} b$ *then* $\mathtt{x}(a) \xrightarrow{*}_{\beta} \mathtt{x}(b)$.
2. *If* $a \to_{Mu_1, Mu_2} b$ *then* $\mathtt{x}(a) \xrightarrow{*}_{\mu} \mathtt{x}(b)$.

As a result, we have that $\to_{\overline{\lambda}\mu\mathtt{x}}$ is a sound refinement of $\to_{\beta,\mu}$.

Corollary 2. *For any pure terms* $a, b \in \mathcal{T}_{\overline{\lambda}\mu\mathtt{x}} \cup \mathcal{L}_{\overline{\lambda}\mu\mathtt{x}}$, *if* $a \xrightarrow{*}_{\overline{\lambda}\mu\mathtt{x}} b$ *then* $a \xrightarrow{*}_{\beta,\mu} b$.

Proof. By Lemmas 3 and 4, and Remark 1. □

It is now easy to show that $\overline{\lambda}\mu\mathtt{x}$-reduction is confluent, using the confluence of β, μ-reduction in the $\lambda\mu$-calculus [15]. The result also holds in the typed case, so the confluence of the local-step cut-elimination procedure follows.

Theorem 3. *The reduction relation* $\to_{\overline{\lambda}\mu\mathtt{x}}$ *is confluent.*

Proof. Since pure terms in $\mathcal{T}_{\overline{\lambda}\mu\mathtt{x}}$ are isomorphic to $\lambda\mu$-terms (Proposition 1 and Theorem 1), $\to_{\beta,\mu}$ is confluent on pure terms in $\mathcal{T}_{\overline{\lambda}\mu\mathtt{x}}$ by the confluence of $\to_{\beta,\mu}$ on $\lambda\mu$-terms. Also, the confluence of $\to_{\beta,\mu}$ on pure terms l in $\mathcal{L}_{\overline{\lambda}\mu\mathtt{x}}$ follows by induction on the structure of l. Now suppose that $a \xrightarrow{*}_{\overline{\lambda}\mu\mathtt{x}} a_1$ and $a \xrightarrow{*}_{\overline{\lambda}\mu\mathtt{x}} a_2$. Then by Lemmas 3 and 4, $\mathtt{x}(a) \xrightarrow{*}_{\beta,\mu} \mathtt{x}(a_i)$ $(i = 1, 2)$, so by the confluence of $\to_{\beta,\mu}$ on pure terms, there is a pure term a' such that $\mathtt{x}(a_i) \xrightarrow{*}_{\beta,\mu} a'$ $(i = 1, 2)$. We also have $a_i \xrightarrow{*}_{\overline{\lambda}\mu\mathtt{x}} \mathtt{x}(a_i)$ by Lemma 2, and $\mathtt{x}(a_i) \xrightarrow{*}_{\overline{\lambda}\mu\mathtt{x}} a'$ by Theorem 2. So we conclude that $a_i \xrightarrow{*}_{\overline{\lambda}\mu\mathtt{x}} a'$ $(i = 1, 2)$. □

6 Strong Normalization

In this section we prove PSN for the $\overline{\lambda}\mu\mathtt{x}$-calculus with respect to β, μ-reduction on pure terms and, at the same time, strong normalization for typed $\overline{\lambda}\mu\mathtt{x}$-terms, which implies strong normalization for the local-step cut-elimination procedure. Following [6, 14], we adapt Bloo and Geuvers' technique [1] to Herbelin-style calculus.

Definition 2 (Bounded terms). *Let* $a \in \mathcal{T}_{\overline{\lambda}\mu\mathtt{x}} \cup \mathcal{L}_{\overline{\lambda}\mu\mathtt{x}}$. *a is said to be* bounded *if for every subterm* b *of* a, $\mathtt{x}(b) \in \mathcal{SN}_{\beta,\mu}$.

Lemma 5. *For any typed* $\overline{\lambda}\mu\mathtt{x}$*-term* a, a *is bounded.*

Proof. Let a be a typed $\overline{\lambda}\mu\mathtt{x}$-term and b be a subterm of a. Then b is also typed, and by subject reduction, so is $\mathtt{x}(b)$. From (the proof of) Proposition 5 and strong normalization of $\to_{\beta,\mu}$ on typed $\lambda\mu$-terms [16], it follows that $\mathtt{x}(b) \in \mathcal{SN}_{\beta,\mu}$. □

Table 5. First-order encoding

$$
\begin{array}{ll}
\mathcal{T}([]) & =_{def} \star \\
\mathcal{T}(u :: l) & =_{def} \mathsf{ii}(\mathcal{T}(u), \mathcal{T}(l)) \\
\mathcal{T}(xl) & =_{def} \mathsf{i}(\mathcal{T}(l)) \\
\mathcal{T}(\lambda x.t) & =_{def} \mathsf{i}(\mathcal{T}(t)) \\
\mathcal{T}(\mu\alpha.[\gamma]t) & =_{def} \mathsf{i}(\mathcal{T}(t)) \\
\mathcal{T}(al) & =_{def} \mathsf{cut}^{|\mathtt{x}(al)|}(\mathcal{T}(a), \mathcal{T}(l)) \\
\mathcal{T}(a\langle v/x\rangle) & =_{def} \mathsf{sub}^{|\mathtt{x}(a\langle v/x\rangle)|}(\mathcal{T}(a), \mathcal{T}(v)) \\
\mathcal{T}(a\langle {}^{[\alpha](-l)}/_{[\alpha]-}\rangle) & =_{def} \mathsf{sub}^{|\mathtt{x}(a\langle {}^{[\alpha](-l)}/_{[\alpha]-}\rangle)|}(\mathcal{T}(a), \mathcal{T}(l))
\end{array}
$$

Definition 3. *Let a be a pure term with $a \in \mathcal{SN}_{\beta,\mu}$. $|a|$ is defined as the maximal length of all β,μ-reduction sequences starting from a.*

Now we encode any bounded term a into a first-order syntax given by the following ordered infinite signature:

$$\star \prec \mathsf{i}(\_) \prec \mathsf{ii}(\_,\_) \prec \mathsf{cut}^n(\_,\_) \prec \mathsf{sub}^n(\_,\_)$$

for all $n \in \mathbb{N}$, and, moreover, $\mathsf{sub}^n(\_,\_) \prec \mathsf{cut}^m(\_,\_)$ for $n < m$. The precedence is well-founded, so the *lexicographic path ordering* (lpo) induced on the first-order terms is also well-founded (definitions and results can be found in [12]). The encoding aforementioned is given in Table 5.

Lemma 6. *If a is bounded and $a \to_{\overline{\lambda}\mu\mathtt{x}} b$, then b is also bounded and $\mathcal{T}(a) >_{\mathsf{lpo}} \mathcal{T}(b)$.*

Proof. By induction on the structure of a. Here we consider a few cases where the reduction is at the root.

(a) $a \equiv (\mu\alpha.[\gamma]t)(u :: l) \to_{Mu_1} (\mu\alpha.[\gamma]t\langle {}^{[\alpha](-(u::[]))}/_{[\alpha]-}\rangle)l \equiv b$ ($\alpha \not\equiv \gamma$). Then

$$
\begin{array}{rcl}
\mathcal{T}(a) & = & \mathcal{T}((\mu\alpha.[\gamma]t)(u :: l)) \\
& = & \mathsf{cut}^{|\mathtt{x}(a)|}(\mathcal{T}(\mu\alpha.[\gamma]t), \mathcal{T}(u :: l)) \\
& = & \mathsf{cut}^{|\mathtt{x}(a)|}(\mathsf{i}(\mathcal{T}(t)), \mathsf{ii}(\mathcal{T}(u), \mathcal{T}(l))) \\
\mathcal{T}(b) & = & \mathcal{T}((\mu\alpha.[\gamma]t\langle {}^{[\alpha](-(u::[]))}/_{[\alpha]-}\rangle)l) \\
& = & \mathsf{cut}^{|\mathtt{x}(b)|}(\mathcal{T}(\mu\alpha.[\gamma]t\langle {}^{[\alpha](-(u::[]))}/_{[\alpha]-}\rangle), \mathcal{T}(l)) \\
& = & \mathsf{cut}^{|\mathtt{x}(b)|}(\mathsf{i}(\mathcal{T}(t\langle {}^{[\alpha](-(u::[]))}/_{[\alpha]-}\rangle)), \mathcal{T}(l)) \\
& = & \mathsf{cut}^{|\mathtt{x}(b)|}(\mathsf{i}(\mathsf{sub}^{|\mathtt{x}(t\langle {}^{[\alpha](-(u::[]))}/_{[\alpha]-}\rangle)|}(\mathcal{T}(t), \mathcal{T}(u :: []))), \mathcal{T}(l)) \\
& = & \mathsf{cut}^{|\mathtt{x}(b)|}(\mathsf{i}(\mathsf{sub}^{|\mathtt{x}(t\langle {}^{[\alpha](-(u::[]))}/_{[\alpha]-}\rangle)|}(\mathcal{T}(t), \mathsf{ii}(\mathcal{T}(u), \star))), \mathcal{T}(l))
\end{array}
$$

where

$$\begin{aligned}|\mathtt{x}(a)| &= |\mathtt{x}((\mu\alpha.[\gamma]t)(u :: l))| \\ &> |\mathtt{x}((\mu\alpha.[\gamma]t\langle^{[\alpha](-(u::[]))}/_{[\alpha]-}\rangle)l)| \\ &= |\mathtt{x}(b)|\end{aligned}$$

and

$$\begin{aligned}|\mathtt{x}(b)| &= |\mathtt{x}((\mu\alpha.[\gamma]t\langle^{[\alpha](-(u::[]))}/_{[\alpha]-}\rangle)l)| \\ &= |\{\mu\alpha.[\gamma]\mathtt{x}(t)[^{[\alpha]\{-\}(\mathtt{x}(u)::[])}/_{[\alpha]-}]\}\mathtt{x}(l)| \\ &\geqslant |\mu\alpha.[\gamma]\mathtt{x}(t)[^{[\alpha]\{-\}(\mathtt{x}(u)::[])}/_{[\alpha]-}]| \\ &= |\mathtt{x}(t)[^{[\alpha]\{-\}(\mathtt{x}(u)::[])}/_{[\alpha]-}]| \\ &= |\mathtt{x}(t\langle^{[\alpha](-(u::[]))}/_{[\alpha]-}\rangle)|.\end{aligned}$$

So we can check that b is bounded and $\mathcal{T}(a) >_{\mathsf{lpo}} \mathcal{T}(b)$.

(b) $a \equiv (\mu\delta.[\alpha]t)\langle^{[\alpha](-l)}/_{[\alpha]-}\rangle \to_{\mathtt{x}} \mu\delta.[\alpha](t\langle^{[\alpha](-l)}/_{[\alpha]-}\rangle l) \equiv b$. Then

$$\begin{aligned}\mathcal{T}(a) &= \mathcal{T}((\mu\delta.[\alpha]t)\langle^{[\alpha](-l)}/_{[\alpha]-}\rangle) \\ &= \mathsf{sub}^{|\mathtt{x}(a)|}(\mathcal{T}(\mu\delta.[\alpha]t), \mathcal{T}(l)) \\ &= \mathsf{sub}^{|\mathtt{x}(a)|}(\mathsf{i}(\mathcal{T}(t)), \mathcal{T}(l)) \\ \mathcal{T}(b) &= \mathcal{T}(\mu\delta.[\alpha](t\langle^{[\alpha](-l)}/_{[\alpha]-}\rangle l)) \\ &= \mathsf{i}(\mathcal{T}(t\langle^{[\alpha](-l)}/_{[\alpha]-}\rangle l)) \\ &= \mathsf{i}(\mathsf{cut}^{|\mathtt{x}(t\langle^{[\alpha](-l)}/_{[\alpha]-}\rangle l)|}(\mathcal{T}(t\langle^{[\alpha](-l)}/_{[\alpha]-}\rangle), \mathcal{T}(l))) \\ &= \mathsf{i}(\mathsf{cut}^{|\mathtt{x}(t\langle^{[\alpha](-l)}/_{[\alpha]-}\rangle l)|}(\mathsf{sub}^{|\mathtt{x}(t\langle^{[\alpha](-l)}/_{[\alpha]-}\rangle)|}(\mathcal{T}(t), \mathcal{T}(l)), \mathcal{T}(l)))\end{aligned}$$

where

$$\begin{aligned}|\mathtt{x}(a)| &= |\mathtt{x}((\mu\delta.[\alpha]t)\langle^{[\alpha](-l)}/_{[\alpha]-}\rangle)| \\ &= |\mu\delta.[\alpha]\mathtt{x}(t\langle^{[\alpha](-l)}/_{[\alpha]-}\rangle l)| \\ &= |\mathtt{x}(t\langle^{[\alpha](-l)}/_{[\alpha]-}\rangle l)| \\ &= |\{\mathtt{x}(t\langle^{[\alpha](-l)}/_{[\alpha]-}\rangle)\}\mathtt{x}(l)| \\ &\geqslant |\mathtt{x}(t\langle^{[\alpha](-l)}/_{[\alpha]-}\rangle)|.\end{aligned}$$

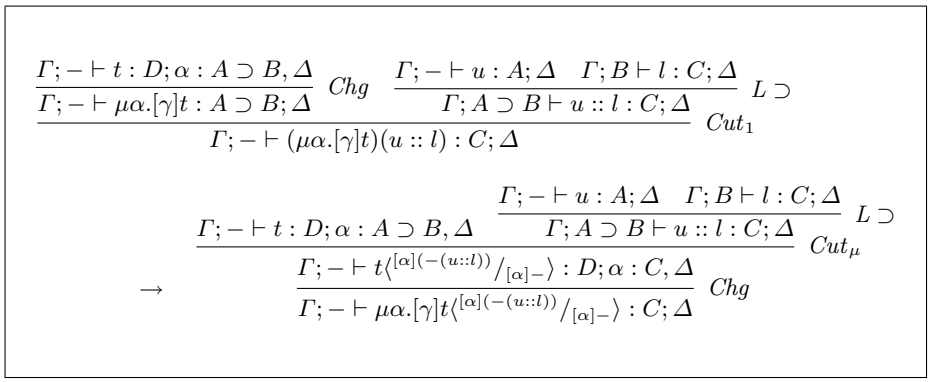
So we can check that b is bounded and $\mathcal{T}(a) >_{\mathsf{lpo}} \mathcal{T}(b)$. □

Theorem 4. *For any bounded term a, $a \in \mathcal{SN}_{\overline{\lambda}\mu\mathtt{x}}$.*

Proof. By the well-foundedness of $>_{\mathsf{lpo}}$ and Lemma 6. □

Corollary 3. *For any typed $\overline{\lambda}\mu\mathtt{x}$-term a, $a \in \mathcal{SN}_{\overline{\lambda}\mu\mathtt{x}}$.*

Fig. 1. Key-case corresponding to (Mu_1^+)



Proof. By Lemma 5 and Theorem 4. □

Corollary 4 (PSN). *For any pure term a, $a \in \mathcal{SN}_{\overline{\lambda}\mu\mathtt{x}}$ if and only if $a \in \mathcal{SN}_{\beta,\mu}$.*

Proof. The only if part is by Corollary 1. Since any pure term $a \in \mathcal{SN}_{\beta,\mu}$ is bounded, we have the if part by Theorem 4. □

7 Variations

In this section we consider some possible variations of $\overline{\lambda}\mu\mathtt{x}$-calculus. As we mentioned in Section 2, the cut-elimination steps corresponding to the (Mu_1) and (Mu_2) rules are not very natural. The reduction rules that correspond to natural cut-elimination steps for Mu_1, Mu_2-redexes are as follows:

$$\begin{array}{ll} (Mu_1^+) & (\mu\alpha.[\gamma]t)(u::l) \to \mu\alpha.[\gamma]t\langle^{[\alpha](-(u::l))}/_{[\alpha]-}\rangle \qquad (\alpha \not\equiv \gamma) \\ (Mu_2^+) & (\mu\alpha.[\alpha]t)(u::l) \to \mu\alpha.[\alpha](t\langle^{[\alpha](-(u::l))}/_{[\alpha]-}\rangle(u::l)). \end{array}$$

In Fig. 1 we show the cut-elimination step corresponding to the (Mu_1^+) rule. For the calculus with these reduction rules instead of the (Mu_1) and (Mu_2) rules, we can also prove results on confluence and strong normalization as we have seen for the $\overline{\lambda}\mu\mathtt{x}$-calculus.

Another, more common approach to a term calculus for classical sequent calculus brings continuation variables $\alpha, \gamma, \ldots$ to the place where we put the empty list []. This leads to two more syntactic categories of contexts and commands which correspond to, in the typing system, sequents without a formula in the stoup on the right side. An advantage of this formulation is the possibility of making substitution for continuation variables the usual substitution. However, to retain the "cut=redex" paradigm, we have to be careful in the combination of left and right rules over the cut rule.

8 Conclusion

Using a term calculus derived from a local-step cut-elimination procedure in Herbelin's sequent calculus, we have given a complete solution to the correspondence problem between normalization and cut-elimination in classical logic. Moreover, we have proved confluence and PSN for the untyped calculus as well as confluence and strong normalization for the local-step cut-elimination procedure.

Since our calculus can be viewed as a conservative extension of $\lambda\mu$-calculus in both term structure and reduction relation, it is useful for conveying substantial work on traditional λ-calculus and $\lambda\mu$-calculus to the world of sequent calculus. We also believe that our analysis can help to design a system in the call-by-value setting, making use of the symmetry inherent in sequent calculus.

Acknowledgements. I was much inspired by Daisuke Kimura and Stéphane Lengrand on the subject of this paper. I also thank Izumi Takeuti and the anonymous referees for valuable comments. This work has been supported by Grant-in-Aid for Scientific Research (17700003) from MEXT of Japan.

References

1. R. Bloo and H. Geuvers. Explicit substitution: On the edge of strong normalization. *Theoretical Computer Science*, 211:375–395, 1999.
2. P.-L. Curien and H. Herbelin. The duality of computation. In *Proceedings of ICFP'00*, pages 233–243. ACM Press, 2000.
3. V. Danos, J.-B. Joinet, and H. Schellinx. LKQ and LKT: Sequent calculi for second order logic based upon dual linear decompositions of classical implication. In *Advances in Linear Logic*, London Mathematical Society Lecture Note Series 222, pages 211–224. Cambridge University Press, 1995.
4. V. Danos, J.-B. Joinet, and H. Schellinx. A new deconstructive logic: Linear logic. *The Journal of Symbolic Logic*, 62:755–807, 1997.
5. R. Dyckhoff and C. Urban. Strong normalisation of Herbelin's explicit substitution calculus with substitution propagation. In *Proceedings of WESTAPP'01*, pages 26–45, 2001.
6. R. Dyckhoff and C. Urban. Strong normalization of Herbelin's explicit substitution calculus with substitution propagation. *Journal of Logic and Computation*, 13:689–706, 2003.
7. J. Espírito Santo. Revisiting the correspondence between cut elimination and normalisation. In *Proceedings of ICALP'00*, Lecture Notes in Computer Science 1853, pages 600–611. Springer-Verlag, 2000.
8. J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.
9. H. Herbelin. A λ-calculus structure isomorphic to Gentzen-style sequent calculus structure. In *Proceedings of CSL'94*, Lecture Notes in Computer Science 933, pages 61–75. Springer-Verlag, 1995.
10. H. Herbelin. *Séquents qu'on calcule*. Thèse de Doctorat, Université Paris 7, 1995.
11. H. Herbelin. Explicit substitutions and reducibility. *Journal of Logic and Computation*, 11:429–449, 2001.

12. S. Kamin and J.-J. Lévy. Attempts for generalizing the recursive path orderings. Handwritten paper, University of Illinois, 1980.
13. S. Lengrand. Call-by-value, call-by-name, and strong normalization for the classical sequent calculus. In *Proceedings of WRS'03*, Electronic Notes in Theoretical Computer Science, 86. Elsevier, 2003.
14. S. Lengrand, R. Dyckhoff, and J. McKinna. A sequent calculus for type theory. To appear in *Proceedings of CSL'06*, Lecture Notes in Computer Science. Springer-Verlag, 2006.
15. M. Parigot. $\lambda\mu$-calculus: An algorithmic interpretation of classical natural deduction. In *Proceedings of LPAR'92*, Lecture Notes in Computer Science 624, pages 190–201. Springer-Verlag, 1992.
16. M. Parigot. Proofs of strong normalisation for second order classical natural deduction. *The Journal of Symbolic Logic*, 62:1461–1479, 1997.
17. E. Polonovski. Strong normalization of $\overline{\lambda}\mu\tilde{\mu}$-calculus with explicit substitutions. In *Proceedings of FoSSaCS'04*, Lecture Notes in Computer Science 2987, pages 423–437. Springer-Verlag, 2004.
18. J. Rocheteau. $\lambda\mu$-calculus and duality: Call-by-name and call-by-value. In *Proceedings of RTA'05*, Lecture Notes in Computer Science 3467, pages 204–218. Springer-Verlag, 2005.
19. C. Urban. *Classical Logic and Computation*. PhD thesis, University of Cambridge, 2000.
20. P. Wadler. Call-by-value is dual to call-by-name. In *Proceedings of ICFP'03*, pages 189–201. ACM Press, 2003.

A Cut-Elimination Steps for Typing Derivations

In this appendix, we display some of the cut-elimination steps corresponding to $\overline{\lambda}\mu$x-reduction for typed terms. First we need the following lemma.

Lemma 7.

1. *If* $\Gamma; \Pi \vdash a : A; \Delta$*, where* x *does not appear in* Γ*, then* $\Gamma, x : B; \Pi \vdash t : A; \Delta$.
2. *If* $\Gamma; \Pi \vdash a : A; \Delta$*, where* α *does not appear in* Δ*, then* $\Gamma; \Pi \vdash t : A; \alpha : B, \Delta$.

Proof. By induction on the structure of derivations. □

Cut-Elimination Steps for Typing Derivations

$(Beta) \quad (\lambda x.t)(u :: l) \rightarrow t\langle u/x\rangle l$

$$\dfrac{\dfrac{\dfrac{\Gamma, x : A; - \vdash t : B; \Delta}{\Gamma; - \vdash \lambda x.t : A \supset B; \Delta}\, R\supset \quad \dfrac{\Gamma; - \vdash u : A; \Delta \quad \Gamma; B \vdash l : C; \Delta}{\Gamma; A \supset B \vdash u :: l : C; \Delta}\, L\supset}{\Gamma; - \vdash (\lambda x.t)(u :: l) : C; \Delta}\, Cut_1}{\rightarrow \quad \dfrac{\dfrac{\Gamma; - \vdash u : A; \Delta \quad \Gamma, x : A; - \vdash t : B; \Delta}{\Gamma; - \vdash t\langle u/x\rangle : B; \Delta}\, Cut_2 \qquad \Gamma; B \vdash l : C; \Delta}{\Gamma; - \vdash t\langle u/x\rangle l : C; \Delta}\, Cut_1}$$

$(Mu_1)\quad (\mu\alpha.[\gamma]t)(u :: l) \to (\mu\alpha.[\gamma]t\langle {}^{[\alpha](-(u::[]))}/_{[\alpha]-}\rangle)l \quad (\alpha \not\equiv \gamma)$

$$\dfrac{\dfrac{\Gamma;- \vdash t : D; \alpha : A \supset B, \Delta}{\Gamma;- \vdash \mu\alpha.[\gamma]t : A \supset B; \Delta}\ Chg \quad \dfrac{\Gamma;- \vdash u : A; \Delta \quad \Gamma; B \vdash l : C; \Delta}{\Gamma; A \supset B \vdash u :: l : C; \Delta}\ L\supset}{\Gamma;- \vdash (\mu\alpha.[\gamma]t)(u :: l) : C; \Delta}\ Cut_1$$

$$\to \quad \dfrac{\dfrac{\dfrac{\Gamma;- \vdash t : D; \alpha : A \supset B, \Delta \quad \Gamma; A \supset B \vdash u :: [] : B; \Delta}{\Gamma;- \vdash t\langle {}^{[\alpha](-(u::[]))}/_{[\alpha]-}\rangle : D; \alpha : B, \Delta}\ Cut_\mu}{\Gamma;- \vdash \mu\alpha.[\gamma]t\langle {}^{[\alpha](-(u::[]))}/_{[\alpha]-}\rangle : B; \Delta}\ Chg \quad \Gamma; B \vdash l : C; \Delta}{\Gamma;- \vdash (\mu\alpha.[\gamma]t\langle {}^{[\alpha](-(u::[]))}/_{[\alpha]-}\rangle)l : C; \Delta}\ Cut_1$$

where $\Gamma; A \supset B \vdash u :: [] : B; \Delta$ is derived from $\Gamma;- \vdash u : A; \Delta$ and the axiom $\Gamma; B \vdash [] : B; \Delta$ by $L\supset$.

$(Mu_2)\quad (\mu\alpha.[\alpha]t)(u :: l) \to (\mu\alpha.[\alpha]t\langle {}^{[\alpha](-(u::[]))}/_{[\alpha]-}\rangle(u :: []))l$

$$\dfrac{\dfrac{\Gamma;- \vdash t : A \supset B; \alpha : A \supset B, \Delta}{\Gamma;- \vdash \mu\alpha.[\alpha]t : A \supset B; \Delta}\ Chg \quad \dfrac{\Gamma;- \vdash u : A; \Delta \quad \Gamma; B \vdash l : C; \Delta}{\Gamma; A \supset B \vdash u :: l : C; \Delta}\ L\supset}{\Gamma;- \vdash (\mu\alpha.[\alpha]t)(u :: l) : C; \Delta}\ Cut_1$$

$$\to \quad \dfrac{\dfrac{\dfrac{\Gamma;- \vdash t : A \supset B; \alpha : A \supset B, \Delta \quad \Gamma; A \supset B \vdash u :: [] : B; \Delta}{\Gamma;- \vdash t\langle {}^{[\alpha](-(u::[]))}/_{[\alpha]-}\rangle : A \supset B; \alpha : B, \Delta}\ Cut_\mu \quad \dfrac{}{\Gamma; A \supset B \vdash u :: [] : B; \alpha : B, \Delta}\ Lemma\ 7}{\Gamma;- \vdash t\langle {}^{[\alpha](-(u::[]))}/_{[\alpha]-}\rangle(u :: []) : B; \alpha : B, \Delta}\ Cut_1}{\Gamma;- \vdash \mu\alpha.[\alpha]t\langle {}^{[\alpha](-(u::[]))}/_{[\alpha]-}\rangle(u :: []) : B; \Delta}\ Chg \quad \Gamma; B \vdash l : C; \Delta}{\Gamma;- \vdash (\mu\alpha.[\alpha]t\langle {}^{[\alpha](-(u::[]))}/_{[\alpha]-}\rangle(u :: []))l : C; \Delta}\ Cut_1$$

$(Musubst_7)\quad (al)\langle {}^{[\alpha](-l')}/_{[\alpha]-}\rangle \to a\langle {}^{[\alpha](-l')}/_{[\alpha]-}\rangle l\langle {}^{[\alpha](-l')}/_{[\alpha]-}\rangle$

$$\dfrac{\dfrac{\Gamma; \Pi \vdash a : B; \alpha : A, \Delta \quad \Gamma; B \vdash l : C; \alpha : A, \Delta}{\Gamma; \Pi \vdash al : C; \alpha : A, \Delta}\ Cut_1 \quad \Gamma; A \vdash l' : D; \Delta}{\Gamma; \Pi \vdash (al)\langle {}^{[\alpha](-l')}/_{[\alpha]-}\rangle : C; \alpha : D, \Delta}\ Cut_\mu$$

$$\to \quad \dfrac{\dfrac{\Gamma; \Pi \vdash a : B; \alpha : A, \Delta \quad \Gamma; A \vdash l' : D; \Delta}{\Gamma; \Pi \vdash a\langle {}^{[\alpha](-l')}/_{[\alpha]-}\rangle : B; \alpha : D, \Delta}\ Cut_\mu \quad \dfrac{\Gamma; B \vdash l : C; \alpha : A, \Delta \quad \Gamma; A \vdash l' : D; \Delta}{\Gamma; B \vdash l\langle {}^{[\alpha](-l')}/_{[\alpha]-}\rangle : C; \alpha : D, \Delta}\ Cut_\mu}{\Gamma; \Pi \vdash a\langle {}^{[\alpha](-l')}/_{[\alpha]-}\rangle l\langle {}^{[\alpha](-l')}/_{[\alpha]-}\rangle : C; \alpha : D, \Delta}\ Cut_1$$

B The $\lambda\mu$-Calculus

The $\lambda\mu$-calculus, introduced in [15], is a λ-calculus extended with control operators. Under the Curry-Howard correspondence, terms of $\lambda\mu$-calculus represent proofs in classical natural deduction. Table 6 presents the syntax and typing rules for simply typed $\lambda\mu$-terms.

The reduction rules of $\lambda\mu$-calculus are the following:

$$\begin{array}{lrcl} (\beta) & (\lambda x.M)N & \to & M[N/x] \\ (\mu) & (\mu\alpha.[\gamma]M)N & \to & \mu\alpha.([\gamma]M)[{}^{[\alpha](-N)}/_{[\alpha]-}]. \end{array}$$

Table 6. $\lambda\mu$-terms and typing rules

$M, N ::= x \mid MN \mid \lambda x.M \mid \mu\alpha.[\gamma]M$

$$\frac{}{\Gamma, x : A \vdash x : A; \Delta}\ Ax \qquad \frac{\Gamma \vdash M : A; \alpha : B, \Delta}{\Gamma \vdash \mu\alpha.[\gamma]M : B; \Delta}\ Chg$$

where $\alpha \not\equiv \gamma$ implies $\gamma : A \in \Delta$

$$\frac{\Gamma \vdash M : A \supset B; \Delta \quad \Gamma \vdash N : A; \Delta}{\Gamma \vdash MN : B; \Delta}\ \supset E \qquad \frac{\Gamma, x : A \vdash M : B; \Delta}{\Gamma \vdash \lambda x.M : A \supset B; \Delta}\ \supset I$$

where $[\_/\_]$ is the usual meta-substitution, and the operation $[^{[\alpha](-N)}/_{[\alpha]-}]$ in the μ-rule replaces inductively each occurrence in $[\gamma]M$ of the form $[\alpha]P$ by $[\alpha](PN)$. In this paper, we consider the β and μ-rules only, and do not include the simplification rules.