Strong Normalization Proof with CPS-Translation for Second Order Classical Natural Deduction

Koji Nakazawa[1], Makoto Tatsuta[2]

[1] Graduate School of Informatics, Kyoto University
e-mail: knak@kuis.kyoto-u.ac.jp
[2] National Institute of Informatics
e-mail: tatsuta@nii.ac.jp

Abstract

This paper points out an error of Parigot's proof of strong normalization of second order classical natural deduction by the CPS-translation, discusses erasing-continuation of the CPS-translation, and corrects that proof by using the notion of augmentations.

1 Introduction

In [7], Parigot gave the typing system $\lambda\mu$-calculus which corresponds to second order classical natural deduction by Curry-Howard isomorphism. He proved strong normalization of that deduction system by proving strong normalization of $\lambda\mu$-calculus. In [7], he proves strong normalization of $\lambda\mu$-calculus in two ways; one uses reducibility method and the other uses the CPS-translation.

Actually his proof by the CPS-translation does not work. He used the property that if $u \rhd_\lambda^1 v$ then $u^* \rhd_\lambda^+ v^*$, which is Proposition 5.1 (i) of [7] and the key of the proof. But that proposition has a counterexample $u = (\mu\alpha.[\beta]x)((\lambda y.y)y)$ and $v = (\mu\alpha.[\beta]x)y$. This is caused by *erasing-continuation* of the CPS-translation: by the CPS-translation the β-redex $(\lambda y.y)y$ in u is transferred to $\mu\alpha.[\beta]x$ through the continuation, but this continuation is erased in $\mu\alpha.[\beta]x$ since α does not occur in $[\beta]x$. By the same kind of errors by erasing-continuation, the strong normalization proof of $\lambda_{exn}^{\rightarrow}$ in [4] and the strong normalization proof of call-by-value symmetric $\lambda\mu$-calculus in [1] do not work either.

In this paper we point out the error of the strong normalization proof of [7], discuss erasing-continuation and preservation of reductions by the CPS-translation, and give a correct proof of strong normalization of $\lambda\mu$-calculus by the CPS-translation. In that proof, we use the notion of augmentations, which translates each $\lambda\mu$-term into a $\lambda\mu$-term of the same meaning in which α necessarily occurs in t for every subterm $\mu\alpha.t$.

2 $\lambda\mu$-calculus

In this section, we define the $\lambda\mu$-calculus, presented in [7]. The $\lambda\mu$-calculus is an extension of the λ-calculus and its type system corresponds to the second-order classical natural deduction.

Definition 2.1. ($\lambda\mu$-calculus)
Terms and reductions of $\lambda\mu$-calculus are defined as follows.

(1) The $\lambda\mu$-calculus has two sorts of variables, which are the λ-variables $x, y \ldots$ and the μ-variables $\alpha, \beta \ldots$. The terms (denoted by $t, u \ldots$) and named-terms (denoted by $e \ldots$) of the $\lambda\mu$-calculus are defined as follows.

$t ::= x \mid (\lambda x.t) \mid (tt) \mid (\mu\alpha.e)$,
$e ::= ([\alpha]t)$.

We call $\lambda x.t$ a λ-abstraction, $t_1 t_2$ an application, $\mu\alpha.e$ a μ-abstraction and $[\alpha]t$ an α-named term. In $\mu\alpha.e$, we consider that the occurrences of α in e are bound. The bound variables may be

renamed as usual when required. $FV(t)$ denotes the set of λ-variables and μ-variables which occur freely in t. We often abbreviate the outermost parentheses and use the following notations,

$\lambda x_1 x_2 \ldots x_n.t \equiv (\lambda x_1.(\lambda x_2.\ldots.(\lambda x_n.t)\ldots))$,

$t_1 t_2 t_3 \ldots t_n \equiv ((\ldots(t_1 t_2)t_3 \ldots)t_n)$.

(2) The reduction rules of the $\lambda\mu$-calculus are the following.

(R_1) $(\lambda x.t)u \rhd^c t[u/x]$,

(R_2) $(\mu\alpha.e)u \rhd^c \mu\alpha.e[u/^*\alpha]$,

The substitution $t[u/x]$ is defined as usual, and the substitution $e[u/^*\alpha]$ is obtained from e by replacing inductively each subterm of the form $[\alpha]v$ by $[\alpha]vu$. The one-step reduction relation $u \rhd^1 v$ holds if and only if v is obtained from u by replacing a subterm u_1 by v_1 with $u_1 \rhd^c v_1$ using (R_1) or (R_2). The one-step λ-reduction $\rhd^1_\lambda$ and the one-step μ-reduction $\rhd^1_\mu$ are similarly defined by only (R_1) and only (R_2) respectively. The strict reduction relations $\rhd^+$, $\rhd^+_\lambda$ and $\rhd^+_\mu$ are the transitive closure of $\rhd^1$, $\rhd^1_\lambda$ and $\rhd^1_\mu$ respectively. The reduction relations $\rhd$, $\rhd_\lambda$ and $\rhd_\mu$ are the reflexive closure of $\rhd^+$, $\rhd^+_\lambda$ and $\rhd^+_\mu$ respectively.

(3) A term u is strongly normalizable if there is no infinite sequence $(u_i)_{i<\omega}$ such that $u \equiv u_0$ and $u_i \rhd^1 u_{i+1}$ for $i < \omega$.

The type assignment system for $\lambda\mu$-calculus is defined as follows.

Definition 2.2. (Second-order type assignment for $\lambda\mu$-calculus)

(1) We use $X, Y \ldots$ for the propositional variables. Types (denoted by $A, B \ldots$) are formulas of second order propositional logic defined as follows.

$A ::= X \mid (A \to A) \mid (\forall X.A)$.

$FV(A)$ denotes the set of free propositional variables in A. We often abbreviate the outermost parentheses.

(2) A λ-context Γ is a finite set of types indexed with λ-variables such that if A^x and B^x are elements of Γ then $A \equiv B$. (Γ, A^x) denotes the union of Γ and $\{A^x\}$. A μ-context Δ is a finite set of types indexed with μ-variables such that if A^α and B^α are elements of Δ then $A \equiv B$. (Δ, A^α) denotes the union of Δ and $\{A^\alpha\}$. A judgement has the form of $t : \Gamma \vdash A, \Delta$, where t is a $\lambda\mu$-term, A is a type, Γ is a λ-context and Δ is a μ-context.

(3) The axioms and rules for the second-order type assignment for $\lambda\mu$-calculus are the following.

$$x : \Gamma, A^x \vdash A, \Delta \quad (ax)$$

$$\frac{t : \Gamma \vdash B, \Delta}{\lambda x.t : \Gamma - \{A^x\} \vdash A \to B, \Delta}\ (\to I) \qquad \frac{t : \Gamma \vdash A \to B, \Delta \quad u : \Gamma \vdash A, \Delta}{tu : \Gamma \vdash B, \Delta}\ (\to E)$$

$$\frac{t : \Gamma \vdash A, \Delta}{t : \Gamma \vdash \forall X.A, \Delta}\ (\forall I) \qquad \frac{t : \Gamma \vdash \forall X.A, \Delta}{t : \Gamma \vdash A[B/X], \Delta}\ (\forall E)$$

$$\frac{t : \Gamma \vdash A, \Delta}{\mu\beta.[\alpha]t : \Gamma \vdash B, (\Delta, A^\alpha) - \{B^\beta\}}\ (\mu)$$

In the rule $(\forall I)$, neither Γ nor Δ contains free X. In the rule (μ), if $C^\alpha \in \Delta$, then $A \equiv C$. A $\lambda\mu$-term t is typable if there exist contexts Γ, Δ and a type A such that $t : \Gamma \vdash A, \Delta$ is provable by the axioms and the rules above.

In [7], the following properties of $\lambda\mu$-calculus were proved.

Proposition 2.3. (Subject Reduction Property)

If $t : \Gamma \vdash A, \Delta$ is provable and $t \rhd u$, then $u : \Gamma \vdash A, \Delta$ is provable.

Proposition 2.4. (Strong Normalization for μ-reduction)

For any $\lambda\mu$-term t_0, there is no infinite sequence $(t_i)_{i<\omega}$ such that $t_i \rhd^1_\mu t_{i+1}$ for all $i < \omega$.

The $\lambda\mu$-calculus enjoys the strong normalization property. In [7], two proofs of the strong normalization were given, one was the reducibility method, and the other used the CPS-translation. The CPS-translation is a map from the $\lambda\mu$-calculus to the λ-calculus, and, from logical point of view, it corresponds to a Kolmogorov translation, which is a map from classical logic to intuitionistic logic.

Definition 2.5.

In the λ-calculus, we fix a λ-variable f and suppose that $\underline{\alpha}$ is a fresh λ-variable for each μ-variable α. The CPS-translation is defined as follows.

(i) $x^* \equiv xf$,
(ii) $(\lambda x.t)^* \equiv f(\lambda xf.t^*)$,
(iii) $(tu)^* \equiv t^*[\lambda z.z(\lambda f.u^*)f/f]$,
(iv) $([\alpha]t)^* \equiv t^*[\underline{\alpha}/f]$,
(v) $(\mu\alpha.e)^* \equiv e^*[f/\underline{\alpha}]$,

where each z is a fresh variable.

The CPS-translation preserves the typability as proved in [7].

Definition 2.6.

Let O be a fixed propositional variable. $\neg_O A$ denotes the type $A \to O$. For any type A, the type A^* is defined as follows.

(i) $X^* \equiv X$,
(ii) $(A \to B)^* \equiv \neg_O\neg_O A^* \to \neg_O\neg_O B^*$,
(iii) $(\forall X.A)^* \equiv \forall X.\neg_O\neg_O A^*$.

For any λ-context $\Gamma = \{A_i^{x_i}; 0 \leq i \leq n\}$, we define $\neg_O\neg_O\Gamma^* = \{(x_i : \neg_O\neg_O A_i^*); 0 \leq i \leq n\}$. For any μ-context $\Delta = \{A_i^{\alpha_i}; 0 \leq i \leq n\}$, we define $\neg_O\Delta^* = \{(\underline{\alpha}_i : \neg_O A_i^*); 0 \leq i \leq n\}$.

Proposition 2.7.

If $t : \Gamma \vdash A, \Delta$ is provable in the second-order type assignment for $\lambda\mu$-calculus then $t^* : \neg_O\neg_O\Gamma^*, \neg_O\Delta^*, (f : \neg_O A^*) \vdash O$ is provable in the second-order λ-calculus.

3 Counterexamples

The CPS-translation preserves also the reduction relation. By this, we can reduce the strong normalization of $\lambda\mu$-calculus to that of the second-order λ-calculus, which has been proved by Girard and whose detailed proof is found in, for example, [2].

However, the proof of [7] does not work for the following reason. The Proposition 5.1 (i) of [7] claims that if $u \rhd_\lambda^1 v$ then $u^* \rhd_\lambda^+ v^*$, but it does not hold because $u \rhd_\lambda^1 v$ and $u^* \equiv v^*$ hold if we take $u \equiv (\mu\alpha.[\beta]x)(Iy)$ and $v \equiv (\mu\alpha.[\beta]x)y$, where I is the term $\lambda x.x$. The difficulty of this method lies in the fact that the CPS-translation does not necessarily preserve the strictness of reductions. For example, as the Proposition 5.1 (ii) of [7], we have $u^* \equiv v^*$ when $u \rhd_\mu v$ even if it is more than one steps. So if t is a μ-abstraction $\mu\alpha.e$ in which e contains no free α, say $t \equiv \mu\alpha.[\beta]x$, then $(tu)^* \equiv t^*$ holds for any term u since $tu \rhd_\mu t$. Therefore, even if $u \rhd_\lambda^+ v$, we have $((\mu\alpha.[\beta]x)u)^* \equiv (\mu\alpha.[\beta]x)^* \equiv ((\mu\alpha.[\beta]x)v)^*$ as the above counterexample. This is caused by erasing-continuation of the CPS-translation: the β-redex in u is transferred to $\mu\alpha.[\beta]x$ through the continuation, but this continuation is erased in $\mu\alpha.[\beta]x$ since α does not occur in $[\beta]x$. Moreover, if we take $(\lambda x.xx)(\lambda x.xx)$ as u, the term $(\mu\alpha.[\beta]x)u$ is not strongly normalizable, while its CPS-translation, identical with $(\mu\alpha.[\beta]x)^*$, is strongly normalizable. Hence, the Proposition 5.4 of [7], which essentially claims the strong normalization, does not hold, so the strong normalization proof with the CPS-translation is not finished.

The strong normalization proof of $\lambda_{exn}^{\to}$, which is an extension of λ-calculus featuring an exception handling mechanism and was presented by de Groote in [4], contains the same kind of error with erasing-continuation. The Proposition 8.3 (a) of [4] claims that, for any expressions M and M' of $\lambda_{exn}^{\to}$, if M is reduced to M' by the rule (β_V) of the modified operational semantics in [4], then we have $M : K \xrightarrow{+}_\beta M' : K$ for any expression K, but it does not hold. For example, $M \equiv$ (`let` $y : \neg\tau$ `in` z `handle` $(y\ x) \Rightarrow Ix$ `end`) is reduced to $M' \equiv$ (`let` $y : \neg\tau$ `in` z `handle`

$(y\ x) \Rightarrow x$ `end`) by the rule (β_V). On the other hand, $M : K$ and $M' : K$ are the same expression Kz. This is caused by erasing-continuation of the CPS-translation: by the CPS-translation the β-redex Ix is transferred to z through the continuation, but this continuation is erased in z since the exception variable y does not occur in z. Furthermore, there is a simpler counterexample `(raise` M`)(`Ix`)` $\rightarrow$ `(raise` M`)`x. For any N, `(raise` M`)`$N : K$ is identical with $M : I$. Therefore, the proof of strong normalizaion of $\lambda^{\rightarrow}_{exn}$ is not finished either.

By erasing-continuation of the CPS-translation, the strong normalization proof of call-by-value symmetric $\lambda\mu$-calculus in [1] does not work either.

4 Strong Normalization Proof with CPS-translation

In this section, we give the correct proof of the strong normalization of $\lambda\mu$-calculus with the CPS-translation. We use the notion of augmentations borrowed from [5] to prevent erasing-continuation of the CPS-translation. By using the notion of augmentations, we translate each $\lambda\mu$-term into a $\lambda\mu$-term of the same meaning in which any μ-abstraction subterm $\mu\alpha.e$ contains a variable α in e.

In the following, we consider the slight extension of the $\lambda\mu$-calculus, in which the named terms are treated as $\lambda\mu$-terms and we can abstract by μ not only named terms but $\lambda\mu$-terms.

Definition 4.1. (Extended $\lambda\mu$-calculus)

Terms of extended $\lambda\mu$-calculus are defined as follows.

$t ::= x \mid (\lambda x.t) \mid (tt) \mid (\mu\alpha.t) \mid ([\alpha]t)$.

Substitutions and reduction relations of extended $\lambda\mu$-calculus are extended from those of $\lambda\mu$-calculus in a straightforward way. Types of extended $\lambda\mu$-calculus are defined as follows.

$A ::= X \mid \bot \mid (A \rightarrow A) \mid (\forall X.A)$.

Typing axioms and rules of extended $\lambda\mu$-calculus are (ax) and four rules for $\rightarrow$ and $\forall$ of $\lambda\mu$-calculus and the following two rules.

$$\frac{t : \Gamma \vdash A, \Delta}{[\alpha]t : \Gamma \vdash \bot, \Delta, A^{\alpha}}\ (\bot I) \qquad \frac{t : \Gamma \vdash \bot, \Delta}{\mu\alpha.t : \Gamma \vdash A, \Delta - \{A^{\alpha}\}}\ (\bot E)$$

In the rule $(\bot I)$, if $B^{\alpha} \in \Delta$, then $A \equiv B$.

Note that, for any $\lambda\mu$-term of the original system t and any type A which contains no $\bot$, if the judgement $t : \Gamma \vdash A, \Delta$ provable in the extended $\lambda\mu$-calculus, then it is provable in original system since the successive application of $(\bot I)$ and $(\bot E)$ is equivalent to the application of (μ). Furthermore, if u is a term of the original system and $u \rhd v$ holds in this extension, then v is a term of the original system and $u \rhd v$ holds in the original system. Therefore, this extension is conservative. The subject reduction property and the strong normalization of μ-reduction for the extended $\lambda\mu$-calculus are proved in a similar way to [7].

Proposition 4.2. (Subject Reduction Property)

If $t : \Gamma \vdash A, \Delta$ is provable in the extended $\lambda\mu$-calculus and $t \rhd u$, then $u : \Gamma \vdash A, \Delta$ is provable in the extended $\lambda\mu$-calculus.

Proposition 4.3. (Strong Normalization for μ-reduction)

For any term t_0 of the extended $\lambda\mu$-calculus, there is no infinite sequence $(t_i)_{i<\omega}$ such that $t_i \rhd^1_{\mu} t_{i+1}$ for all $i < \omega$.

The CPS-translation is extended as follows.

Definition 4.4.

The CPS-translation for the extended $\lambda\mu$-calculus is defined as follows.

(i) $x^* \equiv xf$,

(ii) $(\lambda x.t)^* \equiv f(\lambda xf.t^*)$,

(iii) $(tu)^* \equiv t^*[\lambda z.z(\lambda f.u^*)f/f]$,

(iv) $([\alpha]t)^* \equiv t^*[\underline{\alpha}/f]$,

(v) $(\mu\alpha.t)^* \equiv t^*[I/f][f/\underline{\alpha}]$,

where each z is fresh and $I \equiv \lambda x.x$.

The only difference from the original system is the definition of $(\mu\alpha.t)^*$, but f does not occur freely in t^* if t is a named term, so the definition is the same for the original $\lambda\mu$-terms. It is proved in a straightforward way that the CPS-translation preserves the typability of $\lambda\mu$-terms of the extended system.

Definition 4.5.

$\neg A$ denotes the type $A \to \bot$. For any type A, the type A^* is defined as follows.

(i) $X^* \equiv X$,

(ii) $\bot^* \equiv \bot$,

(iii) $(A \to B)^* \equiv \neg\neg A^* \to \neg\neg B^*$,

(iv) $(\forall X.A)^* \equiv \forall X.\neg\neg A^*$.

For any λ-context $\Gamma = \{A_i^{x_i}; 0 \leq i \leq n\}$, we define $\neg\neg\Gamma^* = \{(x_i : \neg\neg A_i^*); 0 \leq i \leq n\}$. For any μ-context $\Delta = \{A_i^{\alpha_i}; 0 \leq i \leq n\}$, we define $\neg\Delta^* = \{(\underline{\alpha}_i : \neg A_i^*); 0 \leq i \leq n\}$.

Proposition 4.6.

If $t : \Gamma \vdash A, \Delta$ is provable in the second-order type assignment for the extended $\lambda\mu$-calculus then $t^* : \neg\neg\Gamma^*, \neg\Delta^*, (f : \neg A^*) \vdash \bot$ is provable in the second-order λ-calculus.

In the following, we consider only the extended system, since the strong normalization of the original system directly follows from that of the extended system.

We define the augmentations of $\lambda\mu$-terms.

Definition 4.7.

We fix a λ-variable c. For any $\lambda\mu$-term t, we define the set $\mathrm{Aug}(t)$ of terms as follows. We call elements of $\mathrm{Aug}(t)$ augmentations of t.

(i) $\mathrm{Aug}(x) = \{x\}$,

(ii) $\mathrm{Aug}(\lambda x.t) = \{\lambda x.t^+; t^+ \in \mathrm{Aug}(t)\}$,

(iii) $\mathrm{Aug}(tu) = \{t^+u^+; t^+ \in \mathrm{Aug}(t),\ u^+ \in \mathrm{Aug}(u)\}$,

(iv) $\mathrm{Aug}([\alpha]t) = \{[\alpha]t^+; t^+ \in \mathrm{Aug}(t)\}$,

(v) $\mathrm{Aug}(\mu\alpha.t) = \{\mu\alpha.(\lambda z.t^+)([\alpha]c\vec{u});$
$t^+ \in \mathrm{Aug}(t),\ z$ is a fresh λ-variable and $\vec{u}$ is a sequence of terms$\}$,

where $c\vec{u}$ is the application $cu_0u_1 \ldots u_n$ for $\vec{u} \equiv u_0u_1 \ldots u_n$, and $\vec{u}$ may be empty.

Note that, if an augmentation contains a μ-abstraction $\mu\alpha.t$ as its subterm, t always contains an α, except for subterms in the sequence $\vec{u}$ added freely by the definition (v).

Lemma 4.8.

If $t : \Gamma \vdash A, \Delta$ is provable, then there is an augmentation t^+ of t such that $t^+ : \Gamma, (\forall X.X)^c \vdash A, \Delta$.

Proof. This is proved by induction on the derivation of $t : \Gamma \vdash A, \Delta$. We show only the non-trivial case, where the last rule of the derivation is $(\bot E)$. Suppose that the derivation ends with $\dfrac{t : \Gamma \vdash \bot, \Delta, A^\alpha}{\mu\alpha.t : \Gamma \vdash A, \Delta}$. By the induction hypothesis, there is $t^+ \in \mathrm{Aug}(t)$ such that $t^+ : \Gamma, (\forall X.X)^c \vdash \bot, \Delta, A^\alpha$ is provable, so we have $\lambda z.t^+ : \Gamma, (\forall X.X)^c \vdash \neg\bot, \Delta, A^\alpha$. On the other hand, $c : \Gamma, (\forall X.X)^c \vdash A, \Delta$ is provable for any type A, so $[\alpha]c : \Gamma, (\forall X.X)^c \vdash \bot, \Delta, A^\alpha$ is provable. Therefore, we have $(\lambda z.t^+)([\alpha]c) : \Gamma, (\forall X.X)^c \vdash \bot, \Delta, A^\alpha$. Hence $\mu\alpha.(\lambda z.t^+)([\alpha]c) : \Gamma, (\forall X.X)^c \vdash A, \Delta$ is provable, and $\mu\alpha.(\lambda z.t^+)([\alpha]c)$ is an augmentation of $\mu\alpha.t$. □

Lemma 4.9.

(1) For any sequence $\vec{u}$ of terms, we have $f \in FV((c\vec{u})^*)$.

(2) If t is a typable term and is not a named term, then $f \in FV(t^{+*})$ holds for any $t^+ \in \mathrm{Aug}(t)$.

Proof. (1) This is proved by induction on the length of $\vec{u}$. If $\vec{u}$ is empty, then $c^* \equiv cf$ and it contains a free occurrence of f. Suppose that $\vec{u} \equiv \vec{v}w$. Since we have that $(c\vec{v})^*$ contains free f by the induction hypothesis, $(c\vec{v}w)^* \equiv (c\vec{v})^*[\lambda z.z(\lambda f.w^*)f/f]$ contains free f .

(2) This is proved by induction on t. If $t \equiv x$, we have $(x^+)^* \equiv xf$. If $t \equiv \lambda x.u$, the augmentation t^+ of t has the form of $\lambda x.u^+$ for some augmentation u^+ of u. Then we have $t^{+*} \equiv f(\lambda xf.u^{+*})$. If $t \equiv uv$, u is not a named term since uv is typable, and we have $t^{+*} \equiv u^{+*}[\lambda z.z(\lambda f.v^{+*})f/f]$ for some $u^+ \in \mathrm{Aug}(u)$ and $v^+ \in \mathrm{Aug}(v)$. By the induction hypothesis, we have $f \in FV(u^{+*})$, then we have also $f \in FV(t^{+*})$. If $t \equiv \mu\alpha.u$, we may suppose $t^+ \equiv \mu\alpha.(\lambda z.u^+)([\alpha]c\vec{v})$ for some $u^+ \in \mathrm{Aug}(u)$ and sequence $\vec{v}$. Then we have $t^{+*} \equiv (\mu\alpha.(\lambda z.u^+)([\alpha]c\vec{v}))^* \equiv ((\lambda z.z(\lambda f.(c\vec{v})^*[\underline{\alpha}/f])I)(\lambda zf.u^{+*}))[f/\underline{\alpha}]$, and it contains a free occurrence of f since we have $\underline{\alpha} \in FV((c\vec{v})^*[\underline{\alpha}/f])$ from (1). □

Lemma 4.10.

(1) If t and u are $\lambda\mu$-terms, $t^+ \in \mathrm{Aug}(t)$ and $u^+ \in \mathrm{Aug}(u)$, we have $t^+[u^+/x] \in \mathrm{Aug}(t[u/x])$.

(2) If t and u are $\lambda\mu$-terms, $t^+ \in \mathrm{Aug}(t)$ and $u^+ \in \mathrm{Aug}(u)$, we have $t^+[u^+/^*\alpha] \in \mathrm{Aug}(t[u/^*\alpha])$.

Proof. These are proved by induction on t. We show only the non-trivial case of (1) where t is a μ-abstraction $\mu\alpha.v$. By the induction hypothesis, we have $v^+[u^+/x] \in \mathrm{Aug}(v[u/x])$, therefore, for any $\vec{w}$, $(\mu\alpha.(\lambda z.v^+)([\alpha]c\vec{w}))[u^+/x] \equiv \mu\alpha.(\lambda z.v^+[u^+/x])([\alpha](c\vec{w})[u^+/x])$, which is an augmentation of $\mu\alpha.v[u/x]$. Other cases and (2) are similarly proved. □

Lemma 4.11.

(1) For any terms u and v, $u^*[\lambda f.v^*/x] \rhd (u[v/x])^*$.

(2) For any terms u and v, $u^*[\lambda z.z(\lambda f.v^*)\underline{\alpha}/\underline{\alpha}] \equiv (u[v/^*\alpha])^*$.

(3) If $t \rhd^1_\mu u$ then $t^* \equiv u^*$.

(4) If $t \rhd^1_\mu u$ and $t^+ \in \mathrm{Aug}(t)$, then there exists $u^+ \in \mathrm{Aug}(u)$ such that $t^+ \rhd^1_\mu u^+$.

Proof. (1) This is proved by induction on u. When $u \equiv x$, we have LHS$\equiv (\lambda f.v^*)f \rhd^1 v^* \equiv (x[v/x])^* \equiv$RHS. Other cases are proved from induction hypothesis in a straightforward way.

(2) This is proved by induction on u. Suppose that u is an α-named term $[\alpha]t$. Then we have RHS$\equiv$ $([\alpha](t[v/^*\alpha]v))^* \equiv (t[v/^*\alpha])^*[\lambda z.z(\lambda f.v^*)f/f][\underline{\alpha}/f] \equiv t^*[\lambda z.z(\lambda f.v^*)\underline{\alpha}/\underline{\alpha}][\lambda z.z(\lambda f.v^*)f/f][\underline{\alpha}/f]$ by the induction hypothesis. Hence we have RHS$\equiv$ $t^*[\underline{\alpha}/f][\lambda z.z(\lambda f.v^*)\underline{\alpha}/\underline{\alpha}] \equiv$LHS. Other cases are proved from induction hypothesis in a straightforward way.

(3) This is proved by induction on $t \rhd^1_\mu u$. The induction steps are proved in a straightforward way, so we prove only the base case, where t is a redex $(\mu\alpha.v)w$ and $u \equiv \mu\alpha.v[w/^*\alpha]$. In this case, we have $t^* \equiv ((\mu\alpha.v)w)^* \equiv v^*[I/f][f/\underline{\alpha}][\lambda z.z(\lambda f.w^*)f/f]$. Since $FV(v^*[I/f])$ does not contain f, we have $t^* \equiv v^*[I/f][\lambda z.z(\lambda f.w^*)f/\underline{\alpha}]$. Hence we have $t^* \equiv v^*[\lambda z.z(\lambda f.w^*)\underline{\alpha}/\underline{\alpha}][I/f][f/\underline{\alpha}] \equiv (\mu\alpha.v[w/^*\alpha])^* \equiv u^*$ by (2).

(4) This is proved by induction on $t \rhd^1_\mu u$. Similarly, we show only the base case, where $t \equiv (\mu\alpha.v)w$ and $u \equiv \mu\alpha.v[w/^*\alpha]$. In this case, we may suppose $t^+ \equiv (\mu\alpha.(\lambda z.v^+)([\alpha]c\vec{s}))w^+$. Then t^+ is reduced by one-step μ-reduction to $\mu\alpha.(\lambda z.v^+[w^+/^*\alpha])([\alpha]c\vec{s}[w^+/^*\alpha])$, which is an augmentation of u from the lemma 4.10 (2). □

Proposition 4.12.

(1) For any typable term t, if $t \rhd^1_\lambda u$ holds and t^+ is an augmentation of t, then there exists an augmentation u^+ of u such that $t^{+*} \rhd^+ u^{+*}$ holds.

(2) For any term t, if $t \rhd^1_\mu u$ holds and t^+ is an augmentation of t, then there exists an augmentation u^+ of u such that $t^{+*} \equiv u^{+*}$ holds.

Proof. (1) This is proved by induction on $t \rhd^1_\lambda u$.

(Case 1) $t \equiv (\lambda x.t_1)t_2$ and $u \equiv t_1[t_2/x]$. In this case, $t^{+*} \equiv (\lambda z.z(\lambda f.t_2^{+*})f)(\lambda xf.t_1^{+*})$ for some $t_1^+ \in \mathrm{Aug}(t_1)$ and $t_2^+ \in \mathrm{Aug}(t_2)$. Then t^{+*} is reduced to $t_1^{+*}[\lambda f.t_2^{+*}/x]$ by three-step β-reduction. Therefore, we have $t^{+*} \rhd^+ (t_1^+[t_2^+/x])^*$ from the lemma 4.11 (1), and we have $t_1^+[t_2^+/x] \in \mathrm{Aug}(u)$ from the lemma 4.10 (1).

(Case 2) $t \equiv t_1 t_2$, $u \equiv t_1 u_2$ and $t_2 \rhd^1_\lambda u_2$. In this case, t^{+*} has the form of $t_1^{+*}[\lambda z.z(\lambda f.t_2^{+*})f/f]$, and we have $t_2^{+*} \rhd^+ u_2^{+*}$ by the induction hypothesis. Since $t_1 t_2$ is typable, t_1 is not a named term, therefore, t_1^{+*} has a free f from the lemma 4.9 (2). Hence we have $t_1^{+*}[\lambda z.z(\lambda f.t_2^{+*})f/f] \rhd^+ t_1^{+*}[\lambda z.z(\lambda f.u_2^{+*})f/f]$.

Other cases are similarly proved.

(2) This is directly proved from the lemma 4.11 (3) and (4). □

Proposition 4.13. (Strong Normalization of Extended $\lambda\mu$-calculus)
In the extended $\lambda\mu$-calculus, if t is typable term, then t is strongly normalizable.

Proof. Suppose that there exists an infinite sequence of $\lambda\mu$-terms $\{t_i\}_{i<\omega}$ such that t_0 is typable and $t_i \rhd^1 t_{i+1}$ for all $i < \omega$. Note that there are infinitely many i such that $t_i \rhd^1_\lambda t_{i+1}$ from the proposition 4.3. We can find a typable augmentation t_0^+ of t_0 by the proposition 4.8, then t_0^{+*} is also typable by the proposition 4.6. Using the proposition 4.12, there is an infinite sequence of λ-terms $\{t_i^{+*}\}$ such that $t_i^{+*} \rhd t_{i+1}^{+*}$. Then there are infinitely many i such that $t_i^{+*} \rhd^+ t_{i+1}^{+*}$ from the proposition 4.12, but it contradicts the strong normalization of the second-order λ-calculus. □

The strong normalization of the original $\lambda\mu$-calculus directly follows from this proposition.

Theorem 4.14. (Strong Normalization of $\lambda\mu$-calculus)
In the $\lambda\mu$-calculus, if t is typable term, then t is strongly normalizable.

5 Concluding Remarks

The second-order type assignment system considered in this paper has the Curry-style. On the other hand, the Church-style typing system for $\lambda\mu$-calculus was considered, which is the second-order typed $\lambda\mu$-calculus in [7]. The strong normalization for that typed $\lambda\mu$-calculus was also proved by the CPS-translation in [7], but it does not work either by the same error as the proof for the second-order type assignment system. The method of correcting the proof in this paper can be applied to that typed $\lambda\mu$-calculus in a straightforward way.

References

[1] K. Fujita, Domain-free $\lambda\mu$-calculus, Theoretical Informatics and Applications 34 (2000) 433-466.

[2] J.-Y. Girard, P. Taylor and Y. Lafont, Proofs And Types (Cambridge University Press 1989).

[3] P. de Groote, A CPS-translation for the $\lambda\mu$-calculus, *Lecture Notes in Computer Science* **787** (1994) 85-99.

[4] P. de Groote, A Simple Calculus of Exception Handling, *Lecture Notes in Computer Science* **902** (1995) 201-215.

[5] K. Nakazawa, Confluency and Strong Normalizability of Call-by-Value $\lambda\mu$-Calculus, *Theoretical Computer Science*, to appear.

[6] M. Parigot, $\lambda\mu$-Calculus: An Algorithmic Interpretation of Natural Deduction, *Proceedings of the international conference on logic programming and automated reasoning, Lecture Notes in Computer Science* **624** (1992) 190-201.

[7] M. Parigot, Proofs of Strong Normalization for Second Order Classical Natural Deduction, *Journal of Symbolic Logic* **62** (4) (1997) 1461-1479.

[8] G. D. Plotkin, Call-by-Name, Call-by-Value and the λ-calculus, *Theoretical Computer Science* **1** (1975) 125-159.