Strong Normalization Proofs by CPS-translations

Satoshi Ikeda Koji Nakazawa
Graduate School of Informatics, Kyoto University
s-ikeda@kuis.kyoto-u.ac.jp knak@kuis.kyoto-u.ac.jp

Abstract

In this paper, we propose a new proof method for strong normalization of calculi with control operators, and, by this method, we prove strong normalization of the system $\lambda\mu^{\rightarrow\wedge\vee\perp}$, which is introduced in [3] by de Groote and corresponds to the classical natural deduction with disjunctions and permutative conversions by the Curry-Howard isomorphism. For our method, we introduce a new CPS-translation, *continuation and garbage passing style translation* (*CGPS-translation*). Our method by CGPS-translations is applicable to not only $\lambda\mu^{\rightarrow\wedge\vee\perp}$ but many other calculi with control operators such as Parigot's $\lambda\mu$-calculus and its call-by-value variant, and de Groote's calculus $\lambda^{\rightarrow}_{exn}$ for ML-like exception handling. We also show that our method by CGPS-translations gives much simpler proof than any other strong normalization proofs.

Keywords CPS-translation, control operators, strong normalization, classical natural deduction, permutative conversion

1. Introduction

Since Griffin pointed out the correspondence between classical logic and computation with control operators in [6], many typed calculi corresponding classical logic have been introduced, and many authors have studied about computation in calculi with control operators such as `raise` and `handle` for exception handling of ML, `throw`, `call/cc` for first-class continuations of Scheme. These studies are important for both computer science and proof theory, because they give type theoretical views for control mechanisms [2] [10], they make it easier to study classical proof normalization by proof terms [12] [3], and they clarify computational aspects of classical logic [1] [14].

Parigot's $\lambda\mu$-calculus in [11] is one of such typed calculi, which is an extension of λ-calculus and corresponds to classical natural deduction with the rule for reduction to absurdity. Though Parigot introduced $\lambda\mu$-calculus to give proof terms to classical logic, it is also important as abstract programming language with control operators as pointed out by Ong and Stewart. In [10], they introduced a call-by-value variant of the $\lambda\mu$-calculus and a programming language PCF^{-}_{μ} based on the call-by-value $\lambda\mu$-calculus. They also showed that several control operators can be simulated in the call-by-value $\lambda\mu$-calculus. In [3], the typed calculus $\lambda\mu^{\rightarrow\wedge\vee\perp}$ was introduced by de Groote. The system $\lambda\mu^{\rightarrow\wedge\vee\perp}$ is an extension of $\lambda\mu$-calculus and corresponds to the classical natural deduction with conjunctions, disjunctions and permutative conversions.

Strong normalization is one of the most important properties of typed calculi, which guarantees termination of execution of well-typed terms for any evaluation strategy. In [12], Parigot proved the strong normalization of $\lambda\mu$-calculus in two ways, one is the reducibility method, and the other uses the CPS-translation, which translates programs to continuation passing style. By the CPS-translation for the $\lambda\mu$-calculus, which maps $\lambda\mu$-terms to λ-terms and preserves reduction relation and typability, any infinite reduction sequence from any well-typed $\lambda\mu$-term is translated to an infinite reduction sequence from a well-typed λ-term, and it contradicts to the strong normalization of typed λ-calculus. That is, we can reduce strong normalization of $\lambda\mu$-calculus to that of λ-calculus, which is well-known result.

There is, however, difficulty in this CPS-translation method. Nakazawa and Tatsuta discussed about this difficulty for $\lambda\mu$-calculus and its call-by-value variant in [9] and [8]. In many cases, CPS-translations preserving not only convertibility but reduction relation are defined by reducing redexes of control operators in the process of the CPS-translation. Therefore, if a subterm in the source program is eliminated by reduction of control operators, it does not remain in result of the CPS-translation. As for $\lambda\mu$-calculus and its reduction preserving CPS-translation in [12], for example, any *structural reduction* (*μ-reduction*) step is executed in the CPS-translation, that is, if M is reduced to N by structural reduction, then $\overline{M}$ and $\overline{N}$ are identical, where $\overline{M}$ denotes the CPS-translation of M. Therefore, the CPS-translation of $(\mu\alpha.x)N$ is identical with the CPS-translation of $\mu\alpha.x$ for any N, hence any redex in N is erased by the CPS-translation. We call this phenomena *erasing-continuations*. By erasing-continuations, one or more step reduction relations are not necessarily preserved, and that makes it difficult to prove that any infinite reduction sequence is translated to an infinite reduction sequence. As it is pointed out in [9], Parigot's proof by the CPS-translation in [12] contains an error coming from the erasing-continuation problem. Furthermore, the strong normalization proof of $\lambda\mu^{\rightarrow\wedge\vee\perp}$ in [3], which uses the CPS-translation, also contains an error by the erasing-continuation, as pointed out by Matthes [7], so strong normalization proof of $\lambda\mu^{\rightarrow\wedge\vee\perp}$ is not proved.

In this paper, we introduce a new CPS-translation, *continuation and garbage passing style translation* (*CGPS-translation*), and we prove strong normalization of typed calculi with control operators by the CGPS-translation. We show that our method using the CGPS-translations gives much simpler proofs than other methods and is applicable to many calculi.

In CGPS, not only continuations but *garbage terms* are also passed. Garbage terms contain continuations passed in the CPS-translation. Because any part in source program is kept in the CGPS-translated program as continuations in garbage terms, any redex in source program remains as redex. Therefore, CGPS-translation enables us to avoid the erasing-continuation problem and to prove the strong normalization.

This paper is organized as follows. In the sections 2 and 3, we discuss about the strong normalization of the simple $\lambda\mu$-calculus to explain the erasing-continuation problem and the idea of the CGPS-translation. We also show the CGPS-translation method gives much simpler proof for the $\lambda\mu$-calculus. In the section 4, we introduce the CGPS-translation for $\lambda\mu^{\to\wedge\vee\perp}$ and prove the strong normalization. In the section 5, we show that our method is general enough to be applied to strong normalization proof of other calculi with control operators, the call-by-value $\lambda\mu$-calculus and the system $\lambda^{\to}_{exn}$. For the call-by-value $\lambda\mu$-calculus, the strong normalization is proved in [8] by the CPS-translation. We give another proof for the system by the CGPS-translation in this paper, which is much simpler than the proof in [8]. In [2], de Groote introduced a calculus $\lambda^{\to}_{exn}$ for ML-like exception handling and gives a proof of the strong normalization by the CPS-translation. The proof, however, contains an error coming from the erasing-continuation problem, therefore, it has not been proved. We show that the CGPS-translation method can be applied to the strong normalization proof of $\lambda^{\to}_{exn}$.

2. Strong normalization proof by CPS-translation

In this section, we explain the idea and difficulty of the strong normalization proof by CPS-translations, taking the case of the $\lambda\mu$-calculus as an example.

First of all, we give the definition of $\lambda\mu$-calculus. The $\lambda\mu$-calculus is introduced in [11] as term assignment system for second-order classical natural deduction with the rule for reduction to absurdity:

$$\frac{\Gamma;\Delta,\neg\sigma\vdash\perp}{\Gamma;\Delta\vdash\sigma}$$

where σ is a formula Γ is a set of assumptions, Δ is a set of negated assumptions ($\{\neg\sigma_0,\neg\sigma_1\dots\}$). To express these two kinds of assumptions, the $\lambda\mu$-calculus has two kinds of variables, one is λ-variables, which is labels for ordinary assumptions (in Γ), and the other is μ-variables, which is labels for negated assumptions. In this paper, we adopt two-sided notation for judgments for $\lambda\mu$-calculus such that

$$\frac{\Gamma\vdash M:\perp\,;\Delta,\alpha:\sigma}{\Gamma\vdash\mu\alpha.M:\sigma\,;\Delta}$$

for the above rule.

Definition 2.1 ($\lambda\mu$-calculus) The $\lambda\mu$-calculus has two distinct term-variables λ-variables $(x,y,\dots)$ and μ-variables $(\alpha,\beta,\dots)$. Types, terms, singular contexts, typing rules and reduction relations of $\lambda\mu$-calculus are defined as the figure 1.

In λ-abstraction $\lambda x.M$ and μ-abstraction $\mu\alpha.M$, the variables x and α are bound, respectively. The name of bound variables is changed as usual when required.

For a singular context $\mathcal{C}\equiv[]M$, $\mathcal{C}[N]$ denotes the term NM, and for $\mathcal{C}\equiv[]\sigma$, $\mathcal{C}[N]$ denotes $N\sigma$.

For types, X denotes type variables and the negation $\neg\sigma$ is defined as $\neg\sigma\equiv\sigma\to\perp$. In the typing rules, Γ (and Δ) denotes a set of types labeled by λ-variables (and μ-variables) and is called λ-context (and μ-context, respectively). It is not permitted that a context contains two different types labeled by a same variable. Commas between contexts denote union of sets as:

$$\Gamma,\Gamma'\equiv\Gamma\cup\Gamma',\qquad\Gamma,x:\sigma\equiv\Gamma\cup\{x:\sigma\}.$$

The ordinary substitution to λ-variables $M[x:=N]$, to type variables $\sigma[X:=\tau]$ and the renaming of μ-variables $M[\alpha:=\beta]$ are defined as usual. Furthermore, we have the *structural substitution* $M[\alpha\Leftarrow\mathcal{C}]$ defined inductively as follows:

$$\begin{aligned}
x[\alpha\Leftarrow\mathcal{C}]&\equiv x,\\
(\lambda x.M)[\alpha\Leftarrow\mathcal{C}]&\equiv\lambda x.M[\alpha\Leftarrow\mathcal{C}],\\
(M\ N)[\alpha\Leftarrow\mathcal{C}]&\equiv M[\alpha\Leftarrow\mathcal{C}]\ N[\alpha\Leftarrow\mathcal{C}],\\
(\mu\alpha.M)[\alpha\Leftarrow\mathcal{C}]&\equiv\mu\alpha.M,\\
(\mu\beta.M)[\alpha\Leftarrow\mathcal{C}]&\equiv\mu\beta.M[\alpha\Leftarrow\mathcal{C}], &&\text{if }\beta\neq\alpha\\
(\alpha M)[\alpha\Leftarrow\mathcal{C}]&\equiv\alpha\mathcal{C}[M[\alpha\Leftarrow\mathcal{C}]],\\
(\beta M)[\alpha\Leftarrow\mathcal{C}]&\equiv\beta(M[\alpha\Leftarrow\mathcal{C}]), &&\text{if }\beta\neq\alpha\\
(\Lambda X.M)[\alpha\Leftarrow\mathcal{C}]&\equiv\Lambda X.M[\alpha\Leftarrow\mathcal{C}],\\
(M\ \tau)[\alpha\Leftarrow\mathcal{C}]&\equiv M[\alpha\Leftarrow\mathcal{C}]\ \tau.
\end{aligned}$$

The one-step reduction relation $\to$ is the congruence relation defined by the all reduction rules. Similarly, $\to_\beta$ is defined by the rules (β) and (β_t), and $\to_\mu$ is defined by the rule (μ), which is also called *structural reduction*. $\to^*$ is the reflexive transitive closure of $\to$, $\to^+$ is the transitive closure of $\to$, which we call *strict reduction relation*. Similarly $\to^*_\beta$, $\to^+_\beta$, $\to^*_\mu$ and $\to^+_\mu$ are defined.

The rule (μ) is given here in general form, by which, in the following sections, we give several forms of μ-reduction depending on the definition of the singular contexts. For the $\lambda\mu$-calculus, we give only the following two rule schemata:

$$(\mu\alpha M)N\to_\mu\mu\alpha.M[\alpha\Leftarrow[]N],$$

$$(\mu\alpha M)\sigma\to_\mu\mu\alpha.M[\alpha\Leftarrow[]\sigma].$$

The CPS-translation for $\lambda\mu$-calculus, which is a map from $\lambda\mu$-terms to λ-terms, and the negative translation of types are defined as follows. This definition below is essentially same as the translation in [12], though we adopt the definition using colon-notation such as [13] or [2].

Definition 2.2 (Negative translation) The negative translation $\overline{\sigma}$ and the auxiliary translation σ^* for type σ are inductively defined as follows:

$$\begin{aligned}
\overline{\sigma}&\equiv\neg\neg\sigma^*,\\
X^*&\equiv X,\\
\perp^*&\equiv\perp,\\
(\sigma\to\tau)^*&\equiv\overline{\sigma}\to\overline{\tau},\\
\forall X.\sigma^*&\equiv\forall X.\overline{\sigma}.
\end{aligned}$$

Definition 2.3 (CPS-translation) The CPS-translation, which is a map from $\lambda\mu$-term M to λ-term $\overline{M}$, is inductively defined as follows:

$$\begin{aligned}
\overline{M}&\equiv\lambda k.(M:k),\\
x:K&\equiv x\ K,\\
\lambda x.M:K&\equiv K(\lambda x.\overline{M}),\\
MN:K&\equiv M:\lambda m.m\overline{N}K,\\
\mu\alpha.M:K&\equiv(M:\lambda x.x)[k_\alpha:=K],\\
\alpha M:K&\equiv M:k_\alpha,\\
\Lambda X.M:K&\equiv K(\Lambda X.\overline{M}),\\
M\sigma:K&\equiv M:\lambda m.m\sigma^*K,
\end{aligned}$$

where k and k_α for each α are fresh variables in λ-calculus.

From a logical point of view, the negative translation corresponds to one of double negation translations from classical logic to intuitionistic logic, and the CPS-translation corresponds to the translation from classical proof to intuitionistic proof. That is, the CPS-translation preserves typability of terms, which is one of the

Types:

$$\sigma, \tau, \rho ::= X \mid \bot \mid \sigma \to \tau \mid \forall X.\sigma$$

Terms:

$$M, N, L ::= x \mid \lambda x.M \mid M\ N \mid \mu\alpha.M \mid \alpha M \mid \Lambda X.M \mid M\ \sigma$$

Singular contexts:

$$\mathcal{C} ::= []M \mid []\sigma$$

Typing rules:

$$\frac{x\!:\!\tau \in \Gamma}{\Gamma \vdash x\!:\!\tau; \Delta}\ (\text{Ax}) \qquad \frac{\Gamma, x\!:\!\tau \vdash M\!:\!\sigma; \Delta}{\Gamma \vdash \lambda x.M\!:\!\sigma \to \tau; \Delta}\ (\text{Abs}) \qquad \frac{\Gamma \vdash M\!:\!\tau \to \sigma; \Delta \quad \Gamma \vdash N\!:\!\tau; \Delta}{\Gamma \vdash M\ N\!:\!\sigma; \Delta}\ (\text{App})$$

$$\frac{\Gamma \vdash M\!:\!\bot; \Delta, \alpha\!:\!\sigma}{\Gamma \vdash \mu\alpha.M\!:\!\sigma; \Delta}\ (\text{MuAbs}) \qquad \frac{\Gamma \vdash M\!:\!\sigma; \Delta, \alpha\!:\!\sigma}{\Gamma \vdash \alpha M\!:\!\bot; \Delta, \alpha\!:\!\sigma}\ (\text{Nam})$$

$$\frac{\Gamma \vdash M\!:\!\sigma; \Delta}{\Gamma \vdash \Lambda X.M\!:\!\forall X.\sigma; \Delta}\ (\text{AbsTp}) \qquad \frac{\Gamma \vdash M\!:\!\forall X.\sigma; \Delta}{\Gamma \vdash M\ \tau\!:\!\sigma[X\!:=\!\tau]; \Delta}\ (\text{AppTp})$$

Reduction rules:

$$\begin{array}{llll} (\beta) & (\lambda x.M)\ N & \to_\beta & M[x\!:=\!N] \\ (\beta_t) & (\Lambda X.M)\ \sigma & \to_\beta & M[X\!:=\!\sigma] \\ (\mu) & \mathcal{C}[\mu\alpha M] & \to_\mu & \mu\alpha.M[\alpha \Leftarrow \mathcal{C}] \end{array}$$

Figure 1. Definition of $\lambda\mu$-calculus

important properties of the CPS-translation to prove strong normalization. That fact is proved in [12].

Proposition 2.4 (Logical interpretation) *If* $\Gamma \vdash M : \sigma; \Delta$ *is derivable, then the* λ*-term* $\overline{M}$ *is typable with type* $\overline{\sigma}$ *under the context* $\overline{\Gamma}, \neg\Delta^*$*, where*

$$\overline{\Gamma} = \{(x\!:\!\overline{\sigma}) | (x\!:\!\sigma) \in \Gamma\},$$
$$\neg\Delta^* = \{(k_\alpha\!:\!\neg\sigma^*) | (\alpha\!:\!\sigma) \in \Delta\}.$$

Proof. It suffices to prove the following two claims:

(i) $\overline{\Gamma}, \neg\Delta^*, k\!:\!\neg\sigma^* \vdash (M\!:\!k)\!:\!\bot$,

(ii) $\overline{\Gamma}, \neg\Delta^* \vdash \overline{M}\!:\!\overline{\sigma}$.

These are proved by induction on the derivation of $\Gamma \vdash M : \sigma; \Delta$. □

The outline of proof of strong normalization in [12] is as follows: for any $\lambda\mu$-terms M and N, we prove

(i) if $M \to_\beta N$, then $\overline{M} \to^+ \overline{N}$,

(ii) if $M \to_\mu N$, then $\overline{M} \equiv N$,

(iii) $\to_\mu$ is strong normalizable,

therefore, if we have an infinite reduction sequence of typable $\lambda\mu$-term M, there is an infinite reduction sequence of $\overline{M}$, which contradicts to strong normalization of simply typed λ-calculus.

However, (i) does not hold because of *erasing-continuation* problem as it is pointed out in [9]. For example, suppose $M \equiv (\mu\alpha.x)$, then we have $\overline{MN} \equiv \lambda k.x(\lambda x.x)$ for any N. This happens because N is passed to M as continuation $(\lambda m.m\ \overline{N}\ k)$ in CPS of MN, but this continuation is erased since it is not used in M. Therefore, even if N has a β-redex and $MN \to_\beta MN'$ holds, we have $\overline{MN} \equiv \overline{MN'}$.

In [9], Nakazawa and Tatsuta solve this problem by introducing the notion of *augmentations*. Since augmented $\lambda\mu$-terms does not contain μ-abstraction $\mu\alpha.P$ which has no free α in P (such as M in the example above), any continuation is not erased by the CPS-translation. Their method, however, depends on the structure of the $\lambda\mu$-calculus, and it is not applicable to other systems.

In the following sections, we introduce more general solution for erasing-continuation problem, and show it is widely applicable to many other systems.

3. CGPS-translation for $\lambda\mu$-calculus

In this section, we define a new CPS-translation, *continuation and garbage passing style translation* (*CGPS-translation*) for $\lambda\mu$-calculus, and show that we can avoid the erasing-continuation problem by the CGPS-translation and it gives much simpler proof of strong normalization of $\lambda\mu$-calculus. It should be noted that if we ignore the "garbage part" in the definition below, it is entirely same with the definition of the ordinary CPS-translation in the previous section.

Definition 3.1 (Negative translation) The negative translation $\overline{\overline{\sigma}}$ is inductively defined as follows:

$$\begin{aligned} \overline{\overline{\sigma}} &\equiv \top \to \neg\neg\sigma^*, \\ X^* &\equiv X, \\ \bot^* &\equiv \bot, \\ (\sigma \to \tau)^* &\equiv \overline{\overline{\sigma}} \to \overline{\overline{\tau}}, \\ \forall X.\sigma^* &\equiv \forall X.\overline{\overline{\sigma}}, \end{aligned}$$

where $\top$ is the abbreviation of $\bot \to \bot$.

Definition 3.2 (CGPS-translation) The CGPS-translation $\overline{\overline{M}}$ is inductively defined as follows:

$$\begin{aligned}
\overline{\overline{M}} &\equiv \lambda gk.(M : g, k),\\
x : G, K &\equiv x\ G\ K,\\
\lambda x.M : G, K &\equiv \langle\!\langle K\ \lambda x.\overline{\overline{M}}; G\rangle\!\rangle,\\
M\ N : G, K &\equiv (M : \langle\!\langle G; k\rangle\!\rangle, k)[k := \lambda m.m\ \overline{\overline{N}}\ G\ K],\\
\mu\alpha.M : G, K &\equiv (M : g_\alpha, \lambda x.x)[g_\alpha := G][k_\alpha := K],\\
\alpha M : G, K &\equiv \langle\!\langle (M : g_\alpha, k_\alpha); G\rangle\!\rangle,\\
\Lambda X.M : G, K &\equiv \langle\!\langle K\ (\Lambda X.\overline{\overline{M}}); G\rangle\!\rangle,\\
M\ \sigma : G, K &\equiv (M : \langle\!\langle G; k\rangle\!\rangle, k)[k := \lambda m.m\ \sigma^*\ G\ K],
\end{aligned}$$

where $\langle\!\langle M; N\rangle\!\rangle$ denotes $(\lambda x.M)\ N$ with $x \notin \mathit{FV}(M)$, k, g, k_α, g_α for each α are fresh variables in λ-calculus.

The idea of the CGPS-translation is to keep every continuation passed in the CPS-translation in *garbage* (or *trace*). We call G, in the above definition, a garbage term. The CGPS of a $\lambda\mu$-term M has the form of $\lambda gk.(M : g, k)$, in which the parameter g is passed a garbage term similar to k passed a continuation. A garbage term is either a variable or a pair of a garbage term and a continuation$\langle\!\langle G; K\rangle\!\rangle$. It should be noted that we can dispose the second component of the pair, that is, we have $\langle\!\langle G; K\rangle\!\rangle \to G$ for any G and K. The additive $\top$ in the negative translation corresponds to the type of garbage.

In order to simplify the treatment of the structural substitution and structural reduction, we use an abbreviation $\mathcal{K}_\mathcal{C}(G, K)$ for each singular context $\mathcal{C}$ and λ-terms G and K, which is a λ-term defined as

$$\mathcal{K}_\mathcal{C}(G, K) \equiv \begin{cases} \lambda m.m\ \overline{\overline{N}}\ G\ K & \text{if } \mathcal{C} \equiv [\,]\ N,\\ \lambda m.m\ \sigma^*\ G\ K & \text{if } \mathcal{C} \equiv [\,]\ \sigma,\end{cases}$$

where m is a fresh variable. With this abbreviation, $M\ N : G, K$ and $M\ \sigma\ :\ G, K$ can be defined as $\mathcal{C}[M]\ :\ G, K\ \equiv\ M\ :\ \langle\!\langle G; \mathcal{K}_\mathcal{C}(G, K)\rangle\!\rangle, \mathcal{K}_\mathcal{C}(G, K)$ in a uniform manner. This definition makes it clear that the continuation $\mathcal{K}_\mathcal{C}(G, K)$ is kept in the garbage part $\langle\!\langle G; \mathcal{K}_\mathcal{C}(G, K)\rangle\!\rangle$.

This new translation gives a much simpler proof for strong normalization. In the following, we prove:

(i) the CGPS-translation preserves typability,

(ii) the CGPS-translation preserves the strict reduction relation, that is, if $M \to N$ in $\lambda\mu$-calculus then $\overline{\overline{M}} \to^+ \overline{\overline{N}}$.

Therefore, we can immediately show that if typable $\lambda\mu$-term M has an infinite reduction sequence, typable λ-term $\overline{\overline{M}}$ also has infinite reduction sequence.

Firstly, it is proved similarly to the case of the ordinary CPS-translation that the CGPS-translation preserves the typability.

Proposition 3.3 (Logical interpretation) *If $\Gamma \vdash M : \sigma; \Delta$ is derivable, then the λ-term $\overline{\overline{M}}$ is typable with type $\overline{\overline{\sigma}}$ in typed λ-calculus under the context $\overline{\overline{\Gamma}}, \neg\Delta^*, \Delta_\top$, where*

$$\begin{aligned}
\overline{\overline{\Gamma}} &= \{(x : \overline{\overline{\sigma}}) | (x : \sigma) \in \Gamma\},\\
\neg\Delta^* &= \{(k_\alpha : \neg\sigma^*) | (\alpha : \sigma) \in \Delta\},\\
\Delta_\top &= \{(g_\alpha : \top) | (\alpha : \sigma) \in \Delta\}.
\end{aligned}$$

Proof. It suffices to prove the following two claims:

(i) $\overline{\overline{\Gamma}}, \neg\Delta^*, \Delta_\top, g : \top, k : \neg\sigma^* \vdash (M : g, k) : \bot$,

(ii) $\overline{\overline{\Gamma}}, \neg\Delta^*, \Delta_\top \vdash \overline{\overline{M}} : \overline{\overline{\sigma}}$.

These are proved by induction on the derivation of $\Gamma \vdash M : \sigma; \Delta$. In the following, Σ denotes $\overline{\overline{\Gamma}}, \neg\Delta^*, \Delta_\top$. We give proofs for the following two cases of (i).

— In the case where the derivation ends in

$$\frac{\Gamma \vdash M : \bot; \Delta, \alpha : \sigma}{\Gamma \vdash \mu\alpha.M : \sigma; \Delta}\ \text{(MuAbs)}.$$

By the induction hypothesis, we obtain $\Sigma, g_\alpha : \top, k_\alpha : \neg\sigma^*, g : \top, k : \neg\bot^* \vdash (M\ :\ g, k)\ :\ \bot$ with $g, k \notin \mathit{FV}(M)$. Then $\Sigma,\ g : \top\ k : \neg\sigma^* \vdash (M : g_\alpha, \lambda x.x)\ [g_\alpha := g]\ [k_\alpha := k] : \bot$ is derived by the usual substitution lemma, which is easily proved for $\lambda\mu$-calculus.

— In the case where the derivation ends in

$$\frac{\Gamma \vdash M : \sigma; \Delta, \alpha : \sigma}{\Gamma \vdash \alpha M : \bot; \Delta, \alpha : \sigma}\ \text{(Nam)}.$$

By the induction hypothesis, we have $\Sigma, g_\alpha : \top, k_\alpha : \neg\sigma^*, g : \top, k : \neg\sigma^* \vdash (M : g, k) : \bot$ with $g, k \notin \mathit{FV}(M)$. It follows that $\Sigma, g_\alpha : \top, k_\alpha : \neg\sigma^* \vdash (M : g_\alpha, k_\alpha) : \bot$ by the substitution lemma. From the definition of $\langle\!\langle \cdot; \cdot\rangle\!\rangle$, we have $\Sigma, g : \top, k : \neg\bot^*, g_\alpha : \top, k_\alpha : \neg\sigma^* \vdash \langle\!\langle (M : g_\alpha, k_\alpha); g\rangle\!\rangle : \bot$, that is, $\Sigma, g : \top, k : \neg\bot^*, g_\alpha : \top, k_\alpha : \neg\sigma^* \vdash (\alpha M : g, k) : \bot$ as required.

$\square$

Then we prove that the CGPS-translation preserves the strict reduction relation. The key of the proof is that any redex in G remains in the term $M : G, K$, while redexes in K do not necessarily remain in $M : G, K$.

Lemma 3.4 *Let G and K be λ-terms. If $x \in \mathit{FV}(G)$ then $x \in \mathit{FV}(M : G, K)$ for any $\lambda\mu$-term M.*

Proof. By induction on the definition of M. $\square$

Lemma 3.5 *Let M be a $\lambda\mu$-term and N be a λ-term. If $x \notin \mathit{FV}(M)$ then $(M : G, K)[x := N] \equiv M : G[x := N], K[x := N]$.*

Proof. By induction on the definition of M. Note that $\langle\!\langle L_1; L_2\rangle\!\rangle$ $[x := N]$ is identical to $\langle\!\langle L_1[x := N]; L_2[x := N]\rangle\!\rangle$. $\square$

Lemma 3.6 *Let G and G' be λ-terms such that $G \to_\beta G'$. Then $M : G, K \to_\beta^+ M : G', K$ holds.*

Proof. From the lemmas 3.4 and 3.5. $\square$

From this lemma, no β-redex in λ-term G erases in the process of the translation $M\ :\ G, K$. Let us take $(\mu\alpha.x)\ N$, which is the counterexample for the ordinary CPS method in the previous section. We have $\overline{\overline{(\mu\alpha.x) N}} \equiv \lambda gk.(x\ \langle\!\langle g; \lambda m.m\ \overline{\overline{N}}\ g\ k\rangle\!\rangle\ \lambda x.x)$. The garbage $\langle\!\langle g; \lambda m.m\ \overline{\overline{N}}\ g\ k\rangle\!\rangle$ contains the information of N in the continuation $\lambda m.m\ \overline{\overline{N}}\ g\ k$ which is erased by the ordinary CPS-translation. Therefore, if $\overline{\overline{N}} \to_\beta \overline{\overline{N'}}$ holds, then $\overline{\overline{(\mu\alpha.x)\ N}}$ $\to_\beta^+ \overline{\overline{(\mu\alpha.x)\ N'}}$ follows.

Lemma 3.7 *For any $\lambda\mu$-terms M, N and λ-terms G, K, we have*

(i) $(M : G, K)[x := \overline{\overline{N}}] \to_\beta^* M[x := N] : G[x := \overline{\overline{N}}], K[x := \overline{\overline{N}}]$,

(ii) $\overline{\overline{M}}[x := \overline{\overline{N}}] \to_\beta^* \overline{\overline{M[x := N]}}$.

Proof. By induction on the definition of M. $\square$

Lemma 3.8 *Let $\mathcal{C}$ be a singular context, M be a $\lambda\mu$-term, and G and K be λ-terms, then we have*

(i) $(M : G, K)^\diamond \equiv M[\alpha \Leftarrow \mathcal{C}] : G^\diamond, K^\diamond$,

(ii) $\overline{\overline{M}}^{\diamondsuit} \equiv \overline{\overline{M[\alpha \Leftarrow \mathcal{C}]}}$,

where $N^{\diamondsuit}$ *denotes* $N[g_\alpha := \langle\langle g_\alpha; k_\alpha\rangle\rangle][k_\alpha := \mathcal{K}_{\mathcal{C}}(g_\alpha, k_\alpha)]$ *for any* λ*-term* N.

Proof. By induction on the definition of M. We give the proof only for the case of αM.

$$
\begin{aligned}
&(\alpha M : G, K)^{\diamondsuit}\\
&\quad \equiv \langle\langle (M : g_\alpha, k_\alpha); G\rangle\rangle^{\diamondsuit}\\
&\quad \equiv \langle\langle (M : g_\alpha, k_\alpha)^{\diamondsuit}; G^{\diamondsuit}\rangle\rangle\\
&\quad \equiv \langle\langle M[\alpha \Leftarrow \mathcal{C}] : \langle\langle g_\alpha; \mathcal{K}_{\mathcal{C}}(g_\alpha, k_\alpha)\rangle\rangle, \mathcal{K}_{\mathcal{C}}(g_\alpha, k_\alpha); G^{\diamondsuit}\rangle\rangle && (\because \text{IH})\\
&\quad \equiv \langle\langle (M[\alpha \Leftarrow \mathcal{C}] : \langle\langle g_\alpha; k\rangle\rangle, k)[k := \mathcal{K}_{\mathcal{C}}(g_\alpha, k_\alpha)]; G^{\diamondsuit}\rangle\rangle && (k : \text{fresh})\\
&\quad \equiv \langle\langle (\mathcal{C}[M[\alpha \Leftarrow \mathcal{C}]] : g_\alpha, k_\alpha); G^{\diamondsuit}\rangle\rangle\\
&\quad \equiv \alpha\mathcal{C}[M[\alpha \Leftarrow \mathcal{C}]] : G^{\diamondsuit}, K^{\diamondsuit}\\
&\quad \equiv (\alpha M)[\alpha \Leftarrow \mathcal{C}] : G^{\diamondsuit}, K^{\diamondsuit}
\end{aligned}
$$

The other cases are straightforward. □

The CGPS-translation preserves the strict reduction relation, that is to say, one step reduction in $\lambda\mu$-calculus is translated into one or more steps β-reduction in λ-calculus by the CGPS-translation.

Proposition 3.9 *Let* M *and* M' *be* $\lambda\mu$*-terms such that* $M \to M'$. *Then, for any* λ*-terms* G *and* K, *we have*

(i) $M : G, K \to_\beta^{+} M' : G, K$,

(ii) $\overline{\overline{M}} \to_\beta^{+} \overline{\overline{M'}}$.

Proof. By induction on the definition of $M \to M'$.

(β): In the case of β-contraction, the claim is proved by the following equation.

$$
\begin{aligned}
&(\lambda x.M)\ N : G, K\\
&\quad \equiv (\lambda x.M : \langle\langle G; k\rangle\rangle, k)[k := \lambda m.m\ \overline{\overline{N}}\ G\ K]\\
&\quad \equiv \langle\langle k\ \lambda x.\overline{\overline{M}}; \langle\langle G; k\rangle\rangle\rangle\rangle[k := \lambda m.m\ \overline{\overline{N}}\ G\ K]\\
&\quad \to_\beta (k\ \lambda x.\overline{\overline{M}})[k := \lambda m.m\ \overline{\overline{N}}\ G\ K]\\
&\quad \equiv (\lambda m.m\ \overline{\overline{N}}\ G\ K)\ \lambda x.\overline{\overline{M}}\\
&\quad \to_\beta^{+} \overline{\overline{M}}[x := \overline{\overline{N}}]\ G\ K\\
&\quad \to_\beta^{*} \overline{\overline{M[x := N]}}\ G\ K && (\because \text{Lemma 3.7})\\
&\quad \to_\beta^{+} M[x := N] : G, K && (\because \text{Lemma 3.5})
\end{aligned}
$$

The case of (β_t) is proved in the same way.

(μ) and (μ_t): These cases are proved by the lemmas 3.6 and 3.8. Using the notation $\mathcal{K}_{\mathcal{C}}$, We give the proof for them in a uniform manner. We use the notation $N^{\diamondsuit}$ of the lemma 3.8.

$$
\begin{aligned}
&\mathcal{C}[\mu\alpha.M] : G, K\\
&\quad \equiv (\mu\alpha.M : \langle\langle G; k\rangle\rangle, k)[k := \mathcal{K}_{\mathcal{C}}(G, K)]\\
&\quad \equiv (M : g_\alpha, \lambda x.x)[g_\alpha := \langle\langle G; k\rangle\rangle][k_\alpha := k][k := \mathcal{K}_{\mathcal{C}}(G, K)]\\
&\quad \equiv (M : g_\alpha, \lambda x.x)[g_\alpha := \langle\langle G; k_\alpha\rangle\rangle][k_\alpha := \mathcal{K}_{\mathcal{C}}(G, K)]\\
&\quad \equiv (M : g_\alpha, \lambda x.x)^{\diamondsuit}[g_\alpha := G][k_\alpha := K]\\
&\quad \equiv (M[\alpha \Leftarrow \mathcal{C}] : g_\alpha^{\diamondsuit}, \lambda x.x)[g_\alpha := G][k_\alpha := K] && (\because \text{Lemma 3.8})\\
&\quad \to_\beta^{+} (M[\alpha \Leftarrow \mathcal{C}] : g_\alpha, \lambda x.x)[g_\alpha := G][k_\alpha := K]\\
&\quad \equiv \mu\alpha.(M[\alpha \Leftarrow \mathcal{C}]) : G, K
\end{aligned}
$$

Induction step: We give the proof only for the case of $M\ N \to M\ N'$. In this case, we use the lemma 3.6 and the induction hypothesis.

$$
\begin{aligned}
M\ N : G, K &\equiv (M : \langle\langle G; k\rangle\rangle, k)[k := \lambda m.m\ \overline{\overline{N}}\ G\ K]\\
&\to_\beta^{+} (M : \langle\langle G; k\rangle\rangle, k)[k := \lambda m.m\ \overline{\overline{N'}}\ G\ K]\\
&\equiv M\ N' : G, K
\end{aligned}
$$

The other cases are proved in a similar way. □

Theorem 3.10 (Strong normalization) *Any well-typed* $\lambda\mu$*-term is strongly normalizable.*

Proof. The theorem is immediately proved by the propositions 3.3 and 3.9. □

4. Strong normalization of $\lambda\mu^{\to\wedge\vee\bot}$

In this section, we prove the strong normalization of $\lambda\mu^{\to\wedge\vee\bot}$ by CGPS-translation method introduced in the previous section.

The system $\lambda\mu^{\to\wedge\vee\bot}$ is introduced by de Groote in [3], which is an extension of the $\lambda\mu$-calculus and whose type theory corresponds to the classical natural deduction with conjunction, disjunction and permutative conversions by the Curry-Howard isomorphism.

Definition 4.1 ($\lambda\mu^{\to\wedge\vee\bot}$) Types, terms and singular contexts of $\lambda\mu^{\to\wedge\vee\bot}$ is defined as the figure 2. The terms $\mathcal{C}[M]$ for each singular context $\mathcal{C}$, and the structural substitution $M[\alpha \Leftarrow \mathcal{C}]$ are defined similarly to the $\lambda\mu$-calculus. The reductions $\to_D$ and $\to_\delta$ are respectively called *detour reduction* and *permutative conversion*.

In calculi where disjunction is taken as a primitive, the subformula property does not necessarily hold. The permutative conversion $\to_\delta$ is introduced in order to enjoy the subformula property. Indeed, the subformula property of $\lambda\mu^{\to\wedge\vee\bot}$ is proved in [3].

For $\lambda\mu^{\to\wedge\vee\bot}$, de Groote also gives a proof of the strong normalization by the CPS-translation in [3], but Matthes pointed out that the proof contains an error coming from the erasing-continuation problem in [7]. Therefore, the proof of the strong normalization for $\lambda\mu^{\to\wedge\vee\bot}$ in [3] is not finished.

In the following, we prove the strong normalization of $\lambda\mu^{\to\wedge\vee\bot}$ by the CGPS-translation method. The idea of CGPS-translation introduced in the previous section can be applied to $\lambda\mu^{\to\wedge\vee\bot}$ in a straightforward way.

Definition 4.2 (CGPS-translation) The CGPS-translation for $\lambda\mu^{\to\wedge\vee\bot}$ is inductively defined as the figure 3.

As in the case of the $\lambda\mu$-calculus, we use an abbreviation $\mathcal{K}_{\mathcal{C}}(G, K)$ for each singular context $\mathcal{C}$ and λ-terms G and K, which is a λ-term defined as

$$
\mathcal{K}_{\mathcal{C}}(G, K) \equiv \begin{cases}
\lambda m.m\ \overline{\overline{N}}\ G\ K & \text{if } \mathcal{C} \equiv []\ N,\\
\lambda p.p\ (\lambda i_1 i_2.i_j\ G\ K) & \text{if } \mathcal{C} \equiv \pi_j\ [],\\
\lambda m.m\ (\lambda x.(N : G, K))\ (\lambda y.(L : G, K)) & \\
 & \text{if } \mathcal{C} \equiv \delta([], x.N, y.L),
\end{cases}
$$

where m, p, i_1 and i_2 are fresh variables and x and y are not free in G and K. For any singular context $\mathcal{C}$, it can be easily checked that $\mathcal{C}[M] : G, K \equiv M : \langle\langle G; \mathcal{K}_{\mathcal{C}}(G, K)\rangle\rangle, \mathcal{K}_{\mathcal{C}}(G, K)$ as with in $\lambda\mu$-calculus.

Types:

$$\sigma, \tau, \rho ::= X \mid \bot \mid \sigma \to \tau \mid \sigma \wedge \tau \mid \sigma \vee \tau$$

Terms:

$$M, N, L ::= x \mid \lambda x.M \mid M\ N \mid \langle M, N \rangle \mid \pi_j M \mid \iota_j M \mid \delta(M, x.N, y.L) \mid \mu\alpha.M \mid \alpha M \quad (j = 1, 2)$$

Singular contexts:

$$\mathcal{C} ::= []M \mid \pi_j[] \mid \delta([], x.N, y.L)$$

Typing rules:

$$\frac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau; \Delta}\ (\text{Ax}) \qquad \frac{\Gamma, x : \tau \vdash M : \sigma; \Delta}{\Gamma \vdash \lambda x.M : \sigma \to \tau; \Delta}\ (\to\text{-I}) \qquad \frac{\Gamma \vdash M : \tau \to \sigma; \Delta \quad \Gamma \vdash N : \tau; \Delta}{\Gamma \vdash M\ N : \sigma; \Delta}\ (\to\text{-E})$$

$$\frac{\Gamma \vdash M : \bot; \Delta, \alpha : \sigma}{\Gamma \vdash \mu\alpha.M : \sigma; \Delta}\ (\text{MuAbs}) \qquad \frac{\Gamma \vdash M : \sigma; \Delta, \alpha : \sigma}{\Gamma \vdash \alpha M : \bot; \Delta, \alpha : \sigma}\ (\text{Name})$$

$$\frac{\Gamma \vdash M : \sigma; \Delta \quad \Gamma \vdash N : \tau; \Delta}{\Gamma \vdash \langle M, N \rangle : \sigma \wedge \tau; \Delta}\ (\wedge\text{-I}) \qquad \frac{\Gamma \vdash M : \sigma_1 \wedge \sigma_2; \Delta}{\Gamma \vdash \pi_j M : \sigma_j; \Delta}\ (\wedge\text{-E})$$

$$\frac{\Gamma \vdash M : \sigma_j; \Delta}{\Gamma \vdash \iota_j M : \sigma_1 \vee \sigma_2; \Delta}\ (\vee\text{-I}) \qquad \frac{\Gamma \vdash M : \tau \vee \rho; \Delta \quad \Gamma x : \tau \vdash N : \sigma; \Delta \quad \Gamma y : \rho \vdash L : \sigma; \Delta}{\Gamma \vdash \delta(M, x.N, y.L) : \sigma; \Delta}\ (\vee\text{-E})$$

Reduction rules:

$$\begin{array}{clcl}
(\beta_\to) & (\lambda x.M)\ N & \to_\beta & M[x := N] \\
(\beta_\wedge) & \pi_j \langle M_1, M_2 \rangle & \to_\beta & M_j \\
(\beta_\vee) & \delta(\iota_j M, x_1.N_1, x_2.N_2) & \to_\beta & N_j[x_j := M] \\
(\delta) & \mathcal{C}[\delta(M, x.N, y.L)] & \to_\delta & \delta(M, x.\mathcal{C}[N], y.\mathcal{C}[L]) \\
(\mu) & \mathcal{C}[\mu\alpha M] & \to_\mu & \mu\alpha.M[\alpha \Leftarrow \mathcal{C}]
\end{array}$$

Figure 2. Definition of $\lambda\mu^{\to\wedge\vee\bot}$

$$\begin{aligned}
\overline{\overline{M}} &\equiv \lambda gk.(M : g, k) \\
x : G, K &\equiv x\ G\ K \\
\lambda x.M : G, K &\equiv \langle\langle K\ \lambda x.\overline{\overline{M}}; G \rangle\rangle \\
M\ N : G, K &\equiv (M : \langle\langle G; k \rangle\rangle, k)[k := \lambda m.m\ \overline{\overline{N}}\ G\ K] \\
\pi_j M : G, K &\equiv (M : \langle\langle G; k \rangle\rangle, k)[k := \lambda p.p\ (\lambda i_1 i_2.i_j\ G\ K)] \\
\iota_j M : G, K &\equiv \langle\langle K\ (\lambda i_1 i_2.i_j\ \overline{\overline{M}}); G \rangle\rangle \\
\langle M, N \rangle : G, K &\equiv \langle\langle K\ (\lambda p.p\ \overline{\overline{M}}\ \overline{\overline{N}}); G \rangle\rangle \\
\delta(M, x.N, y.L) : G, K &\equiv (M : \langle\langle G; k \rangle\rangle, k)[k := \lambda m.m\ (\lambda x.(N : G, K))\ (\lambda y.(L : G, K))] \\
\mu\alpha.M : G, K &\equiv (M : g_\alpha, \lambda x.x)[g_\alpha := G][k_\alpha := K] \\
\alpha M : G, K &\equiv \langle\langle M : g_\alpha, k_\alpha; G \rangle\rangle
\end{aligned}$$

Figure 3. CGPS-translation for $\lambda\mu^{\to\wedge\vee\bot}$

Definition 4.3 (Negative translation) The negative translation $\overline{\overline{\sigma}}$ for $\lambda\mu^{\to\wedge\vee\bot}$ is inductively defined as follows:

$$\begin{aligned}
\overline{\overline{\sigma}} &\equiv \top \to \neg\neg\sigma^*,\\
X^* &\equiv X,\\
\bot^* &\equiv \bot,\\
(\sigma \to \tau)^* &\equiv \overline{\overline{\sigma}} \to \overline{\overline{\tau}},\\
(\sigma \wedge \tau)^* &\equiv \neg(\overline{\overline{\sigma}} \to \neg\overline{\overline{\tau}}),\\
(\sigma \vee \tau)^* &\equiv \neg\overline{\overline{\sigma}} \to \neg\neg\overline{\overline{\tau}}.
\end{aligned}$$

Proposition 4.4 (Logical interpretation) *If $\Gamma \vdash M:\sigma;\Delta$ is derivable, then the λ-term $\overline{\overline{M}}$ is typable with type $\overline{\overline{\sigma}}$ under the context $\overline{\overline{\Gamma}}, \neg\Delta^*, \Delta_\top$, where*

$$\begin{aligned}
\overline{\overline{\Gamma}} &= \{(x:\overline{\overline{\sigma}})|(x:\sigma) \in \Gamma\},\\
\neg\Delta^* &= \{(k_\alpha:\neg\sigma^*)|(\alpha:\sigma) \in \Delta\},\\
\Delta_\top &= \{(g_\alpha:\top)|(\alpha:\sigma) \in \Delta\}.
\end{aligned}$$

Proof. It suffices to prove the following two claims:

(i) $\overline{\overline{\Gamma}}, \neg\Delta^*, \Delta_\top, g:\top, k:\neg\sigma^* \vdash (M:g,k):\bot$,

(ii) $\overline{\overline{\Gamma}}, \neg\Delta^*, \Delta_\top \vdash \overline{\overline{M}}:\overline{\overline{\sigma}}$.

These are proved by the induction on the derivation of $\Gamma \vdash M:\sigma;\Delta$ in a straightforward way.

□

The proof of the preservation of the strict reduction relation for $\lambda\mu^{\to\wedge\vee\bot}$ is similar to that for the $\lambda\mu$-calculus in the previous section.

Lemma 4.5 *Let G and K be λ-terms. If $x \in FV(G)$ then $x \in FV(M:G,K)$ for any $\lambda\mu^{\to\wedge\vee\bot}$-term M.*

Proof. By induction on the definition of $M:G,K$. □

Lemma 4.6 *Let M be a $\lambda\mu^{\to\wedge\vee\bot}$-term and N be a λ-term. If $x \notin FV(M)$ then $(M:G,K)[x:=N] \equiv M:G[x:=N],K[x:=N]$.*

Proof. By induction on the definition of M. □

Lemma 4.7 *Let G and G' be λ-terms such that $G \to_\beta G'$. Then $M:G,K \to_\beta^+ M:G',K$ holds.*

Proof. From the lemmas 4.5 and 4.6. □

Lemma 4.8 *For any $\lambda\mu^{\to\wedge\vee\bot}$-terms M, N and λ-terms G, K, we have*

(i) $(M:G,K)[x:=\overline{\overline{N}}] \to_\beta^* M[x:=N]:G[x:=\overline{\overline{N}}],K[x:=\overline{\overline{N}}]$,

(ii) $\overline{\overline{M}}[x:=\overline{\overline{N}}] \to_\beta^* \overline{\overline{M[x:=N]}}$.

Proof. By induction on the definition of M. □

Lemma 4.9 *Let $\mathcal{C}$ be a singular context, M be a $\lambda\mu^{\to\wedge\vee\bot}$-term, and G and K be λ-terms, then*

(i) $(M:G,K)^\diamondsuit \equiv M[\alpha\Leftarrow\mathcal{C}]:G^\diamondsuit,K^\diamondsuit$,

(ii) $\overline{\overline{M}}^\diamondsuit \equiv \overline{\overline{M[\alpha\Leftarrow\mathcal{C}]}}$,

where $N^\diamondsuit$ denotes $N[g_\alpha := \langle\langle g_\alpha;k_\alpha\rangle\rangle][k_\alpha := \mathcal{K}_\mathcal{C}(g_\alpha,k_\alpha)]$ for any λ-term N.

Proof. The proof is similar to that of the lemma 3.8. □

Proposition 4.10 *Let M and M' be $\lambda\mu^{\to\wedge\vee\bot}$-terms such that $M \to M'$. Then, for any λ-terms G and K, we have*

(i) $M:G,K \to_\beta^+ M':G,K$,

(ii) $\overline{\overline{M}} \to_\beta^+ \overline{\overline{M'}}$.

Proof. By induction on the definition of $M \to M'$.

Detour reductions ($\to_D$): The case of $(\lambda x.M)\ N \to_D M[x:=N]$ is proved as follows:

$$\begin{aligned}
&(\lambda x.M)\ N:G,K\\
&\quad\equiv (\lambda x.M:\langle\langle G;k\rangle\rangle,k)[k:=\lambda m.m\ \overline{\overline{N}}\ G\ K]\\
&\quad\equiv \langle\langle k\ \lambda x.\overline{\overline{M}};\langle\langle G;k\rangle\rangle\rangle\rangle[k = \lambda m.m\ \overline{\overline{N}}\ G\ K]\\
&\quad\to_\beta (k\ \lambda x.\overline{\overline{M}})[k = \lambda m.m\ \overline{\overline{N}}\ G\ K]\\
&\quad\equiv (\lambda m.m\ \overline{\overline{N}}\ G\ K)\ \lambda x.\overline{\overline{M}}\\
&\quad\to_\beta^+ M[x:=N]:G,K
\end{aligned}$$

The case of $\pi_j\langle M_1,M_2\rangle \to_D M_j$ is proved as follows. Let $\mathcal{C}$ be a singular context $\pi_j[]$.

$$\begin{aligned}
&\pi_j\langle M_1,M_2\rangle:G,K\\
&\quad\equiv (\langle M_1,M_2\rangle:\langle\langle G;k\rangle\rangle,k)[k:=\mathcal{K}_\mathcal{C}(G,K)]\\
&\quad\equiv \langle\langle k\ (\lambda p.p\ \overline{\overline{M_1}}\ \overline{\overline{M_2}});\langle\langle G;k\rangle\rangle\rangle\rangle[k:=\mathcal{K}_\mathcal{C}(G,K)]\\
&\quad\to_\beta \mathcal{K}_\mathcal{C}(G,K)\ (\lambda p.p\ \overline{\overline{M_1}}\ \overline{\overline{M_2}})\\
&\quad\to_\beta^+ \overline{\overline{M_j}}\ G\ K\\
&\quad\to_\beta^+ M_j:G,K
\end{aligned}$$

The case of $\delta(\iota_j M, x_1.N_1, x_2.N_2) \to_D N_j[x_j := M]$ is proved as follows. Let $\mathcal{C}$ be a singular context $\delta([], x_1.N_1, x_2.N_2)$.

$$\begin{aligned}
&\delta(\iota_j M, x_1.N_1, x_2.N_2):G,K\\
&\quad\equiv (\iota_j M:\langle\langle G;k\rangle\rangle,k)[k:=\mathcal{K}_\mathcal{C}(G,K)]\\
&\quad\equiv \langle\langle k\ (\lambda i_1 i_2.i_j\ \overline{\overline{M}});\langle\langle G;k\rangle\rangle\rangle\rangle[k:=\mathcal{K}_\mathcal{C}(G,K)]\\
&\quad\to_\beta (k\ (\lambda i_1 i_2.i_j\ \overline{\overline{M}}))[k:=\mathcal{K}_\mathcal{C}(G,K)]\\
&\quad\equiv \mathcal{K}_\mathcal{C}(G,K)\ (\lambda i_1 i_2.i_j\ \overline{\overline{M}})\\
&\quad\to_\beta^+ (\lambda x_j.(N_j:G,K))\ \overline{\overline{M}}\\
&\quad\to_\beta^+ N_j[x_j:=M]:G,K
\end{aligned}$$

Permutative conversions ($\to_\delta$): The case of permutative conversions is proved as follows. Let M', N' and L' respectively denote $(M:\langle\langle G;k\rangle\rangle,k)$, $\lambda x.(N : \langle\langle G;\kappa\rangle\rangle,\kappa)$ and $\lambda y.(L : \langle\langle G;\kappa\rangle\rangle,\kappa)$.

$$\begin{aligned}
&\mathcal{C}[\delta(M,x.N,y.L)]:G,K\\
&\quad\equiv (\delta(M,x.N,y.L):\langle\langle G;\kappa\rangle\rangle,\kappa)[\kappa:=\mathcal{K}_\mathcal{C}(G,K)]\\
&\quad\equiv (M:\langle\langle\langle\langle G;\kappa\rangle\rangle;k\rangle\rangle,k)[k:=\lambda m.m\ N'\ L'][\kappa:=\mathcal{K}_\mathcal{C}(G,K)]\\
&\quad\to_\beta^+ (M:\langle\langle G;k\rangle\rangle,k)[k:=\lambda m.m\ N'\ L'][\kappa:=\mathcal{K}_\mathcal{C}(G,K)]\\
&\quad\equiv M'[k:=\lambda m.m\ N'[\kappa:=\mathcal{K}_\mathcal{C}(G,K)]\ L'[\kappa:=\mathcal{K}_\mathcal{C}(G,K)]]\\
&\quad\equiv M'[k:=\lambda m.m\ (\lambda x.(\mathcal{C}[N]:G,K))\ (\lambda y.(\mathcal{C}[L]:G,K))]\\
&\quad\equiv \delta(M,x.\mathcal{C}[N],y.\mathcal{C}[L]):G,K
\end{aligned}$$

The case of structural reductions is proved in the same way with the case of (μ) in the proof of the proposition 3.9.

Induction step: It is easily proved by the above lemmas.

□

Theorem 4.11 (Strong normalization) *Any well-typed $\lambda\mu^{\to\wedge\vee\bot}$-term is strongly normalizable.*

Proof. The theorem is proved straightforward by the propositions 4.4 and 4.10. □

5. CGPS-translation method for other calculi

In this section, we show that the CGPS-translations can be applied to strong normalization proofs for other calculi with control operators.

5.1 Strong normalization of call-by-value $\lambda\mu$-calculus

The call-by-value variant of $\lambda\mu$-calculus is firstly introduced by Ong and Stewart in [10], and its strong normalization is discussed by Fujita in [4]. Fujita gives a proof using CPS-translation, but the proof is not finished because of erasing-continuation problem. In [8], this fact is pointed out and the strong normalization is proved by CPS-translation with many auxiliary notions, such as augmentations similar to [9]. The proof in [8] is, however, very complicated and hard to undersand. In contrast, a much simpler proof is given below by the CGPS-translation method.

Definition 5.1 (Call-by-value $\lambda\mu$-calculus) Types and terms of the call-by-value $\lambda\mu$-calculus is the same with those of the $\lambda\mu$-calculus, except we do not consider polymorphic types for simplicity. [1] Values are defined as follows:

$$V, U ::= x \mid \lambda x.M \mid \alpha M.$$

Singular contexts are define as follows:

$$\mathcal{C} ::= V[] \mid []M.$$

Reduction rules are the following:

$$\begin{array}{cccc}(\beta_V) & (\lambda x.M)\,V & \to_\beta & M[x:=V],\\ (\mu) & \mathcal{C}[\mu\alpha M] & \to_\mu & \mu\alpha.M[\alpha\Leftarrow\mathcal{C}],\\ (rn) & \alpha(\mu\beta.V) & \to_s & V[\beta:=\alpha],\\ (\mu\eta) & \mu\alpha.\alpha M & \to_s & M,\end{array}$$

where, in the rule $(\mu\eta)$, M does not contain free α.

The structural reduction for the call-by-value $\lambda\mu$-calculus contains so-called *symmetric structural reduction*, such as

$$V(\mu\alpha.M) \to \mu\alpha.M[\alpha\Leftarrow V[]].$$

By this rule, for any context $\mathcal{E}$ [2], we have

$$\mathcal{E}[\mu\alpha.\cdots\alpha M\cdots] \;\to^*\; \mu\alpha.\cdots\alpha\mathcal{E}[M]\cdots,$$

which can be seen as the α-named subterm M calls the continuation $\mathcal{E}$ of the μ-abstraction. Therefore, in the call-by-value $\lambda\mu$-calculus, we can simulate control operators such as `raise` and `handle` of ML, and `call/cc` of Scheme.

After the definition in [10], the call-by-value $\lambda\mu$-calculus considered here does not have deterministic reduction strategy, that is, we can reduce any part of term unless it goes against the call-by-value principle. Though having such symmetric reduction, this system has good properties as computational system, such as Church-Rosser property and strong normalization, which are stated in [10] and [8].

The idea of CGPS-translation also can be applied to the call-by-value $\lambda\mu$-calculus in a straightforward way, and it gives a much simpler proof of strong normalization, which is similar to those in previous sections.

[1] It is straightforward to extend the discussion here to second-order (domain-free) call-by-value $\lambda\mu$-calculus

[2] Here, *contexts* are defined as

$$\mathcal{E} ::= [] \mid V\mathcal{E} \mid \mathcal{E}M.$$

Definition 5.2 (CGPS-translation) The CGPS-translation $\overline{\overline{M}}$ for $\lambda\mu$-term M and the auxiliary translations $(M : G, K)$ and $\Phi(V)$ (for each value V) are inductively defined as the figure 4. In this definition, P and Q denote terms which are not values.

Definition 5.3 (Negative translation) The negative translation for the call-by-value $\lambda\mu$-calculus is inductively defined as follows:

$$\begin{aligned}\overline{\overline{\sigma}} &\equiv \top \to \neg\neg\sigma^*,\\ X^* &\equiv X,\\ \bot^* &\equiv \bot,\\ (\sigma\to\tau)^* &\equiv \sigma^* \to \overline{\overline{\tau}}.\end{aligned}$$

Proposition 5.4 (Logical interpretation) *If $\Gamma \vdash M : \sigma; \Delta$ is derivable, then the λ-term $\overline{\overline{M}}$ is typable with type $\overline{\overline{\sigma}}$ in typed λ-calculus under the context $\Gamma^*, \neg\Delta^*, \Delta_\top$, where*

$$\begin{aligned}\Gamma^* &= \{(x:\sigma^*) \mid (x:\sigma)\in\Gamma\},\\ \neg\Delta^* &= \{(k_\alpha:\neg\sigma^*) \mid (\alpha:\sigma)\in\Delta\},\\ \Delta_\top &= \{(g_\alpha:\top) \mid (\alpha:\sigma)\in\Delta\}.\end{aligned}$$

Proof. It suffices to prove the following three claims:

(i) $\Gamma^*, \neg\Delta^*, \Delta_\top \vdash \Phi(M) : \sigma^*$, (if M is value),
(ii) $\Gamma^*, \neg\Delta^*, \Delta_\top, g:\top, k:\neg\sigma^* \vdash (M : g, k) : \bot$,
(iii) $\Gamma^*, \neg\Delta^*, \Delta_\top \vdash \overline{\overline{M}} : \overline{\overline{\sigma}}$.

These are proved by induction on the derivation of $\Gamma \vdash M : \sigma; \Delta$. □

Proposition 5.5 *Let M and N be $\lambda\mu$-terms. If $M \to N$ in the call-by-value $\lambda\mu$-calculus then we have $\overline{\overline{M}} \to_\beta^+ \overline{\overline{N}}$.*

Proof. The proposition is proved in a similar way to the calculi in the previous sections, except, for the case of (rn), we need the additional lemma

$$\Phi(V)[g_\beta := g_\alpha][k_\beta := k_\alpha] \equiv \Phi(V[\beta:=\alpha])$$

for any V, which is easily proved. □

Theorem 5.6 (Strong normalization) *Any well-typed $\lambda\mu$-term is strongly normalizable in the call-by-value $\lambda\mu$-calculus.*

Proof. The theorem is proved straightforward by the propositions 5.4 and 5.5. □

5.2 Strong normalization of $\lambda_{exn}^{\to}$

The system $\lambda_{exn}^{\to}$ is introduced by de Groote in [2], which is a typed calculus for ML-like exception handling mechanism with `raise` and `handle`. $\lambda_{exn}^{\to}$ is interesting also from a logical point of view, since the type theory of this system corresponds to the classical natural deduction with the rule for excluded middle:

$$\frac{\Gamma, \tau \vdash \sigma \quad \Gamma, \neg\tau \vdash \sigma}{\Gamma \vdash \sigma}\,.$$

Definition 5.7 ($\lambda_{exn}^{\to}$) Types of $\lambda_{exn}^{\to}$ is the same with the types of $\lambda\mu$-calculus except polymorphic types. $\lambda_{exn}^{\to}$ has two kind of variables, λ-variables (denoted by $x, \ldots$) and *exception variables* (denoted by $y, \ldots$). Terms, values, typing rules and reduction rules of $\lambda_{exn}^{\to}$ are defined as the figure 5.

In the typing rules, Γ is a set in which each element is either a type labeled by a λ-variable or a negated type labeled by an exception variable.

In the reduction rule $(\mathcal{H}/\mathcal{R})$, we use the following abbreviation:

$$\langle (y_i).M \mid (x_i.N_i)\rangle \;\equiv\; \langle y_1.\cdots\langle y_n.M \mid x_n.N_n\rangle\cdots \mid x_1.N_1\rangle.$$

$$
\begin{aligned}
\overline{\overline{M}} &\equiv \lambda gk.(M\!:\!g,k)\\
V\!:\!G,K &\equiv \langle\!\langle K\ \Phi(V);G\rangle\!\rangle\\
V\ W\!:\!G,K &\equiv \Phi(V)\ \Phi(W)\ G\ K\\
V\ Q\!:\!G,K &\equiv (Q\!:\!\langle\!\langle G;k\rangle\!\rangle,k)[k\!:=\!\lambda n.\Phi(V)\ n\ G\ K]\\
P\ W\!:\!G,K &\equiv (P\!:\!\langle\!\langle G;k\rangle\!\rangle,k)[k\!:=\!\lambda m.m\ \Phi(W)\ G\ K]\\
P\ Q\!:\!G,K &\equiv (P\!:\!\langle\!\langle G;k\rangle\!\rangle,k)[k\!:=\!\lambda m.(Q\!:\!\langle\!\langle G;k\rangle\!\rangle,k)[k\!:=\!\lambda n.m\ n\ G\ K]]\\
\mu\alpha.M\!:\!G,K &\equiv (M\!:\!g_\alpha,\lambda x.x)[g_\alpha\!:=\!G][k_\alpha\!:=\!K]\\
\Phi(x) &\equiv x\\
\Phi(\lambda x.M) &\equiv \lambda x.\overline{\overline{M}}\\
\Phi(\alpha M) &\equiv M\!:\!g_\alpha,k_\alpha
\end{aligned}
$$

Figure 4. CGPS-translation for the call-by-value $\lambda\mu$-calculus

Terms:

$$M,N ::= x \mid y \mid \lambda x.M \mid M\ N \mid (\mathcal{R}\ M) \mid \langle y.M|x.N\rangle$$

Values:

$$V,W ::= x \mid y \mid \lambda x.M \mid yV$$

Singular contexts:

$$\mathcal{C} ::= V[] \mid M[] \mid (\mathcal{R}\ [])$$

Typing rules:

$$\frac{x\!:\!\sigma\in\Gamma}{\Gamma\vdash x\!:\!\sigma}\ \text{(Ax)}\qquad \frac{y\!:\!\neg\sigma\in\Gamma}{\Gamma\vdash y\!:\!\neg\sigma}\ \text{(AxExn)}\qquad \frac{\Gamma,x\!:\!\sigma\vdash M\!:\!\tau}{\Gamma\vdash\lambda x.M\!:\!\sigma\to\tau}\ \text{(Abs)}\qquad \frac{\Gamma\vdash M\!:\!\sigma\to\tau\quad \Gamma\vdash N\!:\!\sigma}{\Gamma\vdash M\ N\!:\!\tau}\ \text{(App)}$$

$$\frac{\Gamma\vdash M\!:\!\sigma}{\Gamma\vdash(\mathcal{R}\ M)\!:\!\bot}\ \text{(Raise)}\qquad \frac{\Gamma,x\!:\!\tau\vdash M\!:\!\sigma\quad \Gamma,y\!:\!\neg\tau\vdash N\!:\!\sigma}{\Gamma\vdash\langle y.M|x.N\rangle\!:\!\sigma}\ \text{(Handle)}$$

Reduction rules:

$$
\begin{array}{crcl}
(\beta_V) & (\lambda x.M)\ V & \to & M[x\!:=\!V]\\
(\mathcal{R}) & \mathcal{C}[(\mathcal{R}\ V)] & \to & (\mathcal{R}\ V)\\
(\mathcal{H}_S) & \langle y.V|x.N\rangle & \to & V\quad (\text{if } y\notin \mathrm{FV}(V))\\
(\mathcal{H}/\mathcal{R}) & \langle (y_i).(\mathcal{R}\ (y_j\ V))|(x_i.N_i)\rangle & \to & \langle (y_i).N_j[x_j\!:=\!V]|(x_i.N_i)\rangle\\
(\mathcal{H}) & \mathcal{C}[\langle y.M|x.N\rangle] & \to & \langle y.\mathcal{C}[M]|x.\mathcal{C}[N]\rangle
\end{array}
$$

Figure 5. Definition of $\lambda^{\rightarrow}_{exn}$

The exception variables intuitively express the name of exceptions, and corresponds to μ-variables in $\lambda\mu$-calculus. The term $(\mathcal{R}\ (yV))$ corresponds to the code

`raise (y` V`),`

which raises the exception y with value V, and the term $\langle y.M|x.N\rangle$ corresponds to the code

`let exception y in` M `handle (y x)` $=>N$,

which contains both declaration of the name y of exception and handling part for the exception y. It should be noted that, in $\lambda^{\rightarrow}_{exn}$, since the declaration of exception y must appear with handling part for the exception y, any uncaught exception does not arise as proved in [2].

The reduction $(\mathcal{H})$ corresponds to the structural rules of $\lambda\mu$-calculus.

In [2], de Groote gave a proof of the strong normalization of $\lambda^{\rightarrow}_{exn}$ by a CPS-translation. However, it contains an error coming from the erasing-continuation problem, which is pointed out in [9]. Similar to the case of $\lambda\mu$-calculus, it is claimed in [2] that any reduction step of (β_V) is preserved as one or more β-reduction in λ-calculus, but it is not correct. With the definition of the CPS-translation in [2], $(\mathcal{R}\ M) : K$ is defined as $M : \lambda x.x$, in which K is erased, then, even if $N \to N'$ is a β-reduction step, we have $\overline{\overline{(\mathcal{R}\ M)N}} \equiv \overline{\overline{(\mathcal{R}\ M)N'}}$. Therefore, the strong normalization proof of $\lambda^{\rightarrow}_{exn}$ in [2] does not work.

In the following, we show that our CGPS-translation method can be applied to $\lambda^{\rightarrow}_{exn}$. The negative translation for $\lambda^{\rightarrow}_{exn}$ is the same with the definition 5.3.

Definition 5.8 (CGPS-translation) The CPS-translation $\overline{\overline{M}}$ of any $\lambda^{\rightarrow}_{exn}$-term M and the auxiliary translations $(M : G, K)$ and $\Phi(V)$ (for each value V) are inductively defined as the figure 6.

Proposition 5.9 (Logical interpretation) *If $\Gamma \vdash M : \sigma$ is derivable, then the λ-term $\overline{\overline{M}}$ is typable with type $\overline{\overline{\sigma}}$ under the context Γ^*, where*

$$\Gamma^* = \{(x\!:\!\sigma^*)|(x\!:\!\sigma)\in\Gamma\}\cup\{(y\!:\!\neg\sigma^*)|(y\!:\!\neg\sigma)\in\Gamma\}.$$

$$\begin{aligned}
\overline{\overline{M}} &\equiv \lambda gk.(M:g,k)\\
V:G,K &\equiv \langle\!\langle K\ \Phi(V); G\rangle\!\rangle\\
V\ W:G,K &\equiv \Phi(V)\ \Phi(W)\ G\ K\\
V\ Q:G,K &\equiv (Q:\langle\!\langle G;k\rangle\!\rangle,k)[k:=\lambda n.\Phi(V)\ n\ G\ K]\\
P\ W:G,K &\equiv (P:\langle\!\langle G;k\rangle\!\rangle,k)[k:=\lambda m.m\ \Phi(V)\ G\ K]\\
P\ Q:G,K &\equiv (P:\langle\!\langle G;k\rangle\!\rangle,k)[k:=\lambda m.(Q:\langle\!\langle G;k\rangle\!\rangle,k)[k:=\lambda n.m\ n\ G\ K]]\\
(\mathcal{R}\ M):G,K &\equiv M:G,\lambda x.x\\
\langle x.M|y.N\rangle:G,K &\equiv \langle\!\langle M:G,K;y\rangle\!\rangle[y:=\lambda x.(N:G,K)]\\
\Phi(x) &\equiv x\\
\Phi(y) &\equiv \lambda vgk.\langle\!\langle k\ (y\ v);g\rangle\!\rangle\\
\Phi(\lambda x.M) &\equiv \lambda x.\overline{\overline{M}}\\
\Phi(y\ V) &\equiv y\ \Phi(V)
\end{aligned}$$

Figure 6. CGPS-translation for $\lambda^{\rightarrow}_{exn}$

Proof. The proof is similar to the proposition 5.4. □

Proposition 5.10 *If $\lambda^{\rightarrow}_{exn}$-terms M and M' satisfy $M \rightarrow M'$ by one of the reduction rule R of $\lambda^{\rightarrow}_{exn}$, then we have:*

(i) $\Phi(M) \rightarrow^{\bullet}_{\beta} \Phi(M')$ $\quad (M, M'$: *values*),

(ii) $M:G,K \rightarrow^{\bullet}_{\beta} M':G,K$,

(iii) $\overline{\overline{M}} \rightarrow^{\bullet}_{\beta} \overline{\overline{M'}}$,

where $\rightarrow^{\bullet}_{\beta} = \begin{cases} \rightarrow^{*}_{\beta} & (R = \mathcal{R} \text{ or } \mathcal{H})\\ \rightarrow^{+}_{\beta} & (R = \beta_V, \mathcal{H}_S \text{ or } \mathcal{H}/\mathcal{R}) \end{cases}$

Proof. The proposition is proved in a similar way to the calculi in the previous sections. □

Though the CGPS-translation for $\lambda^{\rightarrow}_{exn}$ preserves not only convertibility but also reduction relation similarly to the calculi in the previous sections, it does not necessarily preserve strict reduction relations as the proposition 5.10. For example, we have $(\mathcal{R}\ (\mathcal{R}\ M)) \rightarrow (\mathcal{R}\ M)$ by the rule $(\mathcal{R})$, but the CGPS-translation of the two terms are identical to $\lambda gk.(M:g,(\lambda x.x))$. Therefore, we need the following fact, which is proved in [2] by an auxiliary notion of the *norm* of terms.

Proposition 5.11 *The reduction defined by union of $\mathcal{R}$ and $\mathcal{H}$ is strongly normalizable.*

Proof. In [2], a natural number $|M|$ is defined for each term M and it is proved that if $M \rightarrow M'$ holds by either $\mathcal{R}$ or $\mathcal{H}$, then $|M| > |M|'$. □

Theorem 5.12 (Strong normalization) *Any well-typed $\lambda^{\rightarrow}_{exn}$-term is strongly normalizable.*

Proof. The theorem is proved straightforward by the propositions 5.9, 5.10 and 5.11. □

6. Concluding remarks

In this paper, we introduce a new CPS-translation, which we call continuation and garbage passing style (CGPS-translation), and we prove strong normalization of typed calculi with control operators by the CGPS-translation.

The strong normalization of the classical natural deduction $\lambda\mu^{\rightarrow\wedge\vee\perp}$, which has disjunctions and permutative conversion, is newly proved by the CGPS-translation. We also show this method gives much simpler proofs for strong normalization of $\lambda\mu$-calculus and its call-by-value variant. Furthermore, we prove the strong normalization of $\lambda^{\rightarrow}_{exn}$, for which strong normalization has not been proved.

There are several method without using CPS-translations for strong normalization of calculi with control operators. The well-known reducibility candidate (or saturated set) method is applicable to $\lambda\mu$-calculus (proved by Parigot in [12]) and symmetric $\lambda\mu$-calculus (proved by Yamagata in [15]). In [7], Matthes gives another translation from $\lambda\mu$-calculus to λ-calculus, using the notion of the least stable super type. These methods are powerful enough, but they are complicated, and it is hard to extend or apply these ideas to additional features such as additive logical connectives, permutative conversions or other control operators. On the other hand, the idea of our method using the CGPS-translations is much simpler than other methods and it is easy to apply to other systems if only ordinary CPS-translation for the system is defined.

In [5], Gørtz et al. survey several method for reducing strong normalization of λ-calculus to weak normalization of λ-calculus, and they pointed out that these can be characterized by Klop's ι-translation:

$$\iota(\lambda x.P) \equiv \lambda x.[\iota(P),x],$$

by which every β-reduction makes a copy of the argument

$$\iota(\lambda x.[P,x])Q \rightarrow [P[x:=Q],Q],$$

hence, we can avoid erasing arguments. The CGPS-translation, which is introduced to avoid erasing-continuation, is based on similar idea. Garbage terms play the same role with the copy of arguments in ι-translation. It is, however, not a trivial application of the ideas in [5], since in the CGPS-translations, not only we copy continuations as garbage, but pass the garbage together with continuations.

Recently, CPS-translations play important roles to clarify the computational meaning of classical logic with duality. While the systems treated in this paper, such as $\lambda\mu$-calculus and $\lambda\mu^{\rightarrow\wedge\vee\perp}$, are term assignment for natural deduction style classical logic, several symmetric calculi corresponding to sequent calculus style classical logic are studied, for example, Curien and Herbelin's $\overline{\lambda}\mu\tilde{\mu}$ in [1] and Wadler's dual calculus [14]. In these calculi, the duality of classical logic is reflected as the duality of computation between call-by-value and call-by-name, and the CPS-translations for each aspect of computation are considered to clarify their computational meaning. We can expect that these CPS-translations can be extended to CGPS-translations and, by these translations, the strong normalization of these calculi is proved, which is one of the important future works. However, the full reduction systems of these symmetric calculi are non-deterministic and it seems difficult to define CPS-translation for such systems, therefore, we confess that it seems difficult to apply our method to the strong normalization proof for these full reduction systems.

We think it important to study about computational meaning of the CGPS-translations. Though we introduce the CGPS-translations as a solution of the erasing-continuation problem to prove strong normalization the CGPS-translations preserve almost all of strict reduction relations as mentioned in the previous sections. That is, any one or more steps reduction is preserved as a one or more steps reduction relation. It seems that this fact show the CGPS-translations enable to simulate reduction steps of control operations by some corresponding reduction steps in calculi without control operators, and that the CGPS-translations may have some interesting computational meaning. The CGPS-translations, however, map reduction steps of control operation to "garbage disposal" steps. For example, consider the structural reduction step

$$(\mu\alpha.\alpha x)y \to \mu\alpha.\alpha(xy)$$

in $\lambda\mu$-calculus. The CGPS of these terms are,

$$\begin{aligned}\overline{\overline{(\mu\alpha.\alpha x)y}} &\equiv \lambda gk.((\mu\alpha.\alpha x)y):g,k \\ &\equiv \lambda gk.\langle\!\langle x\langle\!\langle g;K\rangle\!\rangle K;\langle\!\langle g;K\rangle\!\rangle\rangle\!\rangle, \\ \overline{\overline{\mu\alpha.\alpha(xy)}} &\equiv \lambda gk.(\mu\alpha.\alpha(xy)):g,k \\ &\equiv \lambda gk.\langle\!\langle xy:g,k;g\rangle\!\rangle \\ &\equiv \lambda gk.\langle\!\langle x\langle\!\langle g;K\rangle\!\rangle K;g\rangle\!\rangle,\end{aligned}$$

where $K \equiv \lambda m.m\overline{\overline{y}}gk$, hence, the β-reduction step $\overline{\overline{(\mu\alpha.\alpha x)y}} \to \overline{\overline{\mu\alpha.\alpha(xy)}}$ is garbage disposal step as follows (the β-redex and its contractum are underlined):

$$\lambda gk.\langle\!\langle x\langle\!\langle g;K\rangle\!\rangle K;\underline{\langle\!\langle g;K\rangle\!\rangle}\rangle\!\rangle \to \lambda gk.\langle\!\langle x\langle\!\langle g;K\rangle\!\rangle K;\underline{g}\rangle\!\rangle.$$

This reduction step disposes the continuation K in garbage $\langle\!\langle g;K\rangle\!\rangle$, and we can not assert that this garbage disposal step simulates the structural reduction. It is a future work to study about CPS-translations which simulate control features in programming languages without control operators, such as λ-calculus, with respect to not only convertibility but reduction steps.

Acknowledgments

We would like to thank Masahiko Sato and Atsushi Igarashi for their helpful comments.

References

[1] Pierre-Louis Curien and Hugo Herbelin. The Duality of Computation. In *International Conference on Functional Programming '00*, 2000.

[2] Philippe de Groote. A simple calculus of exception handling. In M. Dezani-Ciancaglini and G. Plotkin, editors, *Typed Lambda Calculi and Applications, TLCA'95,*, volume 902 of *Lecture Notes in Computer Science*, pages 201–215. 1995.

[3] Philippe de Groote. Strong normalization of classical natural deduction with disjunction. In S. Abramsky, editor, *TLCA 2001*, volume 2044 of *Lecture Notes in Computer Science*, pages 182–196. 2001.

[4] Ken-etsu Fujita. Domain-free $\lambda\mu$-calculus. *Theoretical Informatics and Applications*, 34(6):433–466, 2000.

[5] Inge Li Gørtz, Signe Reuss, and Morten Heine Sørensen. Strong normalization from weak normalization by translation into the lambda-i-calculus. *Higher-Order and Symbolic Computation*, 16:253–285, 2003.

[6] Timothy G. Griffin. A formulae-as-types notion of control. In *Proceedings of POPL '90*, pages 47–58, 1990.

[7] Ralph Matthes. Stabilization — an alternative to double-negation translation for classical natural deduction. In *Proceedings of Logic Colloquium 2003*, 2003.

[8] Koji Nakazawa. Confluency and strong normalizability of call-by-value $\lambda\mu$-calculus. *Theoretical Computer Science*, 290:429–463, 2003.

[9] Koji Nakazawa and Makoto Tatsuta. Strong normalization proof with cps-translation for second order classical natural deduction. *The Journal of Symbolic Logic*, 68(3):851–859, 2003. Corrigendum of this article is available in, *The Journal of Symbolic Logic*, 68(4):1415–1416, 2003.

[10] C.-H. L. Ong and C. A. Stewart. A curry-howard foundation for functional computation with control. In *Proceedings of POPL '97*, pages 215–227, 1997.

[11] Michel Parigot. $\lambda\mu$-calculus: an algorithmic interpretation of classical natural deduction. In *Proceedings of the International Conference on Logic Programming and Automated Reasoning*, volume 624 of *Lecture Notes in Computer Science*, pages 190–201. 1992.

[12] Michel Parigot. Strong normalization for second order classical natural deduction. *The Journal of Symbolic Logic*, 62(4):1461–1479, 1997.

[13] Gordon Plotkin. Call-by-name, call-by-value and the λ-calculus. *Theoretical Computer Science*, 1:125–159, 1975.

[14] Philip Wadler. Call-by-value is dual to call-by-name. In *International Conference on Functional Programming '03*, 2003.

[15] Yoriyuki Yamagata. Strong Normalization of Second Order Symmetric Lambda-mu Calculus. In N. Kobayashi and B.C. Pierce, editors, *Theoretical Aspects of Computer Software, TACS 2001*, volume 2215 of *Lecture Notes in Computer Science*, pages 459–467. 2001.