Logical Methods in Computer Science
Volume 17, Issue 1, 2021, pp. 18:1–18:20
https://lmcs.episciences.org/

Submitted Jul. 10, 2017
Published Mar. 02, 2021

COMPLETE CALL-BY-VALUE CALCULI OF CONTROL OPERATORS, II: STRONG TERMINATION

RYU HASEGAWA

Graduate School of Mathematical Sciences, The University of Tokyo, Komaba 3-8-1, Meguro-ku, Tokyo 153-8914, Japan
e-mail address: ryu@ms.u-tokyo.ac.jp

Abstract. We provide characterizations of the strong termination property of the CCV (complete call-by-value) $\lambda\mu$-calculus introduced in the first part of this series of papers. The calculus is complete with respect to the standard continuation-passing style (CPS) semantics. The union-intersection type system for the calculus was developed in the first paper. We characterize the strong normalizability of the calculus in terms of the CPS semantics and typeability.

Introduction

This is the second half of a series of papers. In the first part, we proposed a call-by-value $\lambda\mu$-calculus, called the CCV $\lambda\mu$-calculus, which is complete for the continuation-passing style (CPS) semantics [Has15]. Furthermore, we proposed the union-intersection type discipline. Among others, we verified the following:

(1) A term M terminates with respect to the call-by-value evaluation if and only if its CPS translation $[\![M]\!]$ is solvable.
(2) A term M is weakly normalizing if and only if its CPS translation $[\![M]\!]$ is weakly normalizing.
(3) A term M terminates with respect to the call-by-value evaluation if and only if M is typeable.
(4) A term M is weakly normalizing if and only if M is typeable where the typing judgment of M contains neither empty intersection nor empty union.

The theme of the second part is to further exntend these results. We give characterizations of the strong termination property. Specifically, we show that

(5) M is strongly normalizing if and only if its CPS translation $[\![M]\!]$ is strongly normalizing.
(6) M is strongly normalizing if and only if M is typeable using empty intersection or empty union nowhere.

After a brief introduction to the CCV $\lambda\mu$-calculus and its type system containing union and intersection, we first verify the strong termination of a fragment in §2.1 as an intermediate step. Employing this result, we verify (6) in Thm. 2.30 and (5) in Thm. 2.31 in §2.2.

Key words and phrases: $\lambda\mu$-calculus, control operators, call-by-value, normalization, type discipline.

LOGICAL METHODS IN COMPUTER SCIENCE
DOI:10.23638/LMCS-17(1:18)2021

An essential idea behind our calculus is a departure from the conventional demand that terms be freely generated by syntactic grammars. An analogy is found in arithmetic. We frequently use expressions such as $3+5+7$. When we calculate it (or implement a calculator on a computer), we forcibly interpret the expression as either $(3+5)+7$ or $3+(5+7)$ and perform the calculation. However, strictly distinguishing between the two cases presents little advantage for humans to understand the essence of arithmetic. We positively use the expression $3+5+7$ as an amalgamation of two types of bracketed expressions, or even as a sum of three numbers. This type of intended ambiguity helps us process arithmetic flexibly. In the same vein, we introduce ambiguity in the constructors of call-by-value calculus. The resulting calculus is complete with respect to the standard semantics, and yet usable.

1. Preliminaries

We recall the CCV $\lambda\mu$-calculus and the union-intersection type discipline for the calculus [Has15]. We also review some of the results needed later. Details are found in the first paper.

1.1. CCV $\lambda\mu$-calculus.

The CCV $\lambda\mu$-calculus is a variant of the call-by-value $\lambda\mu$-calculus. It employs the let-syntax as in Moggi's λ_c-calculus [Mog88]. However, we write the let-binding to the right of its body:

$$M \restriction x := N \qquad \text{in place of} \qquad \mathsf{let}\ x = N\ \mathsf{in}\ M.$$

Note that the order of M and N is reversed.

Formally, the syntax of the CCV $\lambda\mu$-calculus is given as follows. We distinguish ordinary variables and continuation variables. We also distinguish terms M and jumps J. They are defined mutually recursively by the following syntax:

$$\begin{aligned} M &::= x \mid \lambda x.\, M \mid MM \mid M \restriction x := M \mid \mu k.\, J \\ J &::= [k]M \mid J \restriction x := M \end{aligned}$$

where x ranges over ordinary variables and k over continuation variables. The notion of free variables is naturally defined. The let-construct $M \restriction x := N$ binds x, and its scope is M. We use the notation $z \in M$ to denote that a variable z that is ordinary or continuation occurs freely in M.

A key idea is that this syntax is not regarded to freely generate the entities. We introduce syntactic equalities by two associativity axioms:

$$\begin{aligned} L \restriction x := (M \restriction y := N) &= (L \restriction x := M) \restriction y := N && \text{if } y \notin L \\ [k](L \restriction x := M) &= ([k]L) \restriction x := M \end{aligned}$$

where L, M, and N are terms. We do not syntactically distinguish two terms (or two jumps) if they turn out to be equal by a series of application of these rules. We mostly omit brackets:

$$\begin{gathered} L \restriction x := M \restriction y := N \\ \mu k.\, J \restriction x := M \\ [k]L \restriction x := M. \end{gathered}$$

For the first, if the side condition is not satisfied (i.e., if $y \in L$), we take it to mean $(L \restriction x := M) \restriction y := N$. For the second, we read it as $\mu k.\, (J \restriction x := M)$.

A *value* is either a variable or a lambda abstraction. We have the following ten reduction rules, where N is a non-value and V is a value:

$$\begin{array}{lllll}
(ad_1) & NM & \to\ zM \restriction z := N & \\
(ad_2) & VN & \to\ Vz \restriction z := N & \\
(\beta_\lambda) & (\lambda x.\, M)V & \to\ M \restriction x := V & \\
(\beta_{let}) & M \restriction x := V & \to\ M\{V/x\} & \\
(\beta_\mu) & M \restriction x := \mu k.\, J & \to\ \mu k.\, J\{[k]\Box \mapsto [k]M \restriction x := \Box\} & \\
(\beta_{jmp}) & [l]\mu k.\, J & \to\ J\{l/k\} & \\
(\eta_\lambda) & \lambda x.\, Vx & \to\ V & (\text{if } x \notin V) \\
(\eta_{let}) & x \restriction x := M & \to\ M & \\
(\eta_\mu) & \mu k.\, [k]M & \to\ M & (\text{if } k \notin M) \\
(exch) & (\mu k.\, J) \restriction x := M & \to\ \mu k.\, J \restriction x := M & (\text{if } k \notin M)
\end{array}$$

In the first two rules, z is a fresh ordinary variable. We use braces for substitution. The notation $J\{[k]\Box \mapsto [k]M \restriction x := \Box\}$ is a standard context substitution in the $\lambda\mu$-calculus. We write $L =_{ccv} M$ if two terms are equivalent regarding the smallest equivalence relation generated from reductions.

Since we are allowed to alter the scope of let-binding by the equality axioms, the continuation $M \restriction x := \Box$ captured by rule β_μ is changeable, depending on the choice of the scope. This ambiguity is intended and is crucial to verifying the sharpened completeness theorem, which is a key result in our previous paper.

Remark 1.1. In the previous paper, we had the third equality axiom:

$$(\mu k.\, J) \restriction x := M \ =\ \mu k.\, (J \restriction x := M) \qquad \text{if } k \notin M.$$

Namely, we were able to exchange the μ-operator and let-operator. As we commented in the first paper, however, this equality axiom was inessential for proving the sharpened completeness theorem. Instead, we can consider a reduction rule. For the purpose of this paper, the latter approach gives better results at this stage. Therefore, we adopt the rule *exch* above. It is open whether the same results are obtained if the equality axiom is chosen. See Rem. 2.12.

The semantics of the CCV $\lambda\mu$-calculus is given by the call-by-value CPS translation. It maps each CCV term and each jump to a lambda term. In place of a standard CPS translation, we adopt the definition via the colon translation, which is introduced to diminish a number of superfluous redexes [Plo75]. The latter behaves better in regard to reductions, which are our concern. We use the notation $(\!|M|\!)[K]$ in place of $M : K$ for readability.

$$\begin{array}{lll}
(\!|V|\!)[K] & := & KV^* \\
(\!|V_1V_2|\!)[K] & := & V_1^*V_2^*K \\
(\!|VN|\!)[K] & := & (\!|N|\!)[\lambda y.\, V^*yK] \\
(\!|NV|\!)[K] & := & (\!|N|\!)[\lambda x.\, xV^*K] \\
(\!|N_1N_2|\!)[K] & := & (\!|N_1|\!)[\lambda x.\, (\!|N_2|\!)[\lambda y.\, xyK]] \\
(\!|L \restriction x := M|\!)[K] & := & (\!|M|\!)[\lambda x.\, (\!|L|\!)[K]] \\
(\!|\mu k.\, J|\!)[K] & := & (\lambda k.\, (\!|J|\!))K \\
(\!|[k]M|\!) & := & (\!|M|\!)[k] \\
(\!|J \restriction x := M|\!) & := & (\!|M|\!)[\lambda x.\, (\!|J|\!)] \\
x^* & := & x
\end{array}$$

$$(\lambda x.\, M)^* := \lambda x k.\, (\!|M|\!)[k]$$

$$[\![M]\!] := \lambda k.\, (\!|M|\!)[k]$$

Here V and V_i are values, while N and N_i are non-values. L and M are terms. The results are applicable to the standard CPS translation, with some extra arguments, as discussed in Rem. 2.33.

It is better to regard the target of the CPS translation as a sorted lambda calculus. There are four sorts. The terms of the sorted calculus are defined as follows:

Term	T	$::= \lambda k.\, Q \mid WW$
Jump	Q	$::= KW \mid TK$
Value	W	$::= x \mid \lambda x.\, T$
Continuation	K	$::= k \mid \lambda x.\, Q$

The CPS translation yields terms that are subject to this syntax. $(\!|M|\!)[K]$ and $(\!|J|\!)$ produce terms of sort Q, and V^* of sort W. $[\![M]\!]$ has sort T.

We also have the inverse translation $(\text{-})^{-1}$ from the target calculus back into the CCV $\lambda\mu$-calculus. We do not, however, need the concrete shape of the translation, for we use it only through Lem. 1.2 and 1.3 below. We refer the interested reader to [Has15].

We list several results that are needed in this paper from [Has15]. We call a reduction by rule (η_μ) *vertical*, and a reduction by (ad_1) or (ad_2) *administrative*. A non-administrative reduction is called *practical*.

Lemma 1.2. *Let M be a term of the CCV $\lambda\mu$-calculus. There is a term $M^\dagger$ of the calculus such that $M \xrightarrow{A^*} M^\dagger \xleftarrow{V^*} [\![M]\!]^{-1}$ (notice the direction). Here A^* denotes a finite number of administrative reductions and V^* a finite number of vertical reductions.*

Lemma 1.3. *If $P \to_{\beta\eta} Q$ in the target calculus, then $P^{-1} \xrightarrow{+} Q^{-1}$ by one or more steps of practical reductions with no use of rule exch.*

Lemma 1.4. *If $L' \xleftarrow{V^*} L \xrightarrow{P^*} M$ by finite steps of vertical reductions and practical reductions, there is a term M' such that $L' \xrightarrow{P^*} M' \xleftarrow{V^*} M$. Moreover, the arrow $L' \xrightarrow{P^*} M'$ may be an identity only when $L \xrightarrow{P^*} M$ consists solely of β_μ, β_{jmp}, η_μ, and exch.*

In the verifications of Lem. 1.2 and 1.3, we need the equality axioms of our calculus. Sharpened completeness is an immediate consequence of these lemmata. This explains why we include the equality axioms in the CCV $\lambda\mu$-calculus. These lemmata are also necessary to prove one direction of implications in the main theorems.

1.2. **Union-intersection type discipline.** Types are divided into three categories: raw types R, subsidiary types S, and types T. These are defined by the following syntax:

$$R ::= \alpha \mid S \to T$$
$$S ::= \bigcap R$$
$$T ::= \bigcup S$$

where α ranges over atomic types. $\bigcap R$ means a nonempty finite formal intersection $R_1 \cap R_2 \cap \cdots \cap R_n$ $(n \geq 1)$. $\bigcup S$ is similar. Intersection and union follow associativity and commutativity.

Remark 1.5. In our previous work [Has15], the empty intersection ω and the empty union $\curlywedge$ are allowed. In this paper, we use only the type derivations that contain ω or $\curlywedge$ nowhere. So we omit them from the beginning.

We define the subtype relation $\leq$ by the following derivation rules:

$$\frac{}{\alpha \leq \alpha} \qquad \frac{S' \leq S \qquad T \leq T'}{S \to T \leq S' \to T'}$$

$$\frac{S \leq S'}{S \cap S'' \leq S'} \qquad \frac{[\ S \leq S_i\ ]_i}{S \leq \bigcap_i S_i}$$

$$\frac{T \leq T'}{T \leq T'' \cup T'} \qquad \frac{[\ T_i \leq T\ ]_i}{\bigcup_i T_i \leq T}.$$

The notation $[\ S \leq S_i\ ]_i$ means a sequence of derivations where i ranges over a finite index set. $[\ T_i \leq T\ ]_i$ is similar.

A *typing judgment* has the form $\Gamma \vdash M : T \mid \Delta$ where Γ is a finite sequence of $x_i : S_i$, and Δ is a finite sequence of $k_j : T_j$. Note that ordinary variables have only subsidiary types. We assume a special type $\perp\!\!\!\perp$ for typing jumps. The inference rules are given as follows:

$$\frac{}{\Gamma, x : S \vdash x : S \mid \Delta}$$

$$\frac{[\Gamma, x : S_i \vdash M : T_i \mid \Delta]_i}{\Gamma \vdash \lambda x.\, M : \bigcap_i (S_i \to T_i) \mid \Delta}$$

$$\frac{\Gamma \vdash M : \bigcup_i \bigcap_j (S_{ij} \to T) \mid \Delta \qquad [\Gamma \vdash N : \bigcup_j S_{ij} \mid \Delta]_i}{\Gamma \vdash MN : T \mid \Delta}$$

$$\frac{[\Gamma, x : S_i \vdash M : T \mid \Delta]_i \qquad \Gamma \vdash N : \bigcup_i S_i \mid \Delta}{\Gamma \vdash M \upharpoonright x := N : T \mid \Delta}$$

$$\frac{\Gamma \vdash J : \perp\!\!\!\perp \mid \Delta, k : T}{\Gamma \vdash \mu k.\, J : T \mid \Delta}$$

$$\frac{\Gamma \vdash M : T \mid \Delta, k : T}{\Gamma \vdash [k]M : \perp\!\!\!\perp \mid \Delta, k : T}$$

$$\frac{[\Gamma, x : S_i \vdash J : \perp\!\!\!\perp \mid \Delta]_i \qquad \Gamma \vdash N : \bigcup_i S_i \mid \Delta}{\Gamma \vdash J \upharpoonright x := N : \perp\!\!\!\perp \mid \Delta}$$

$$\frac{\Gamma \vdash M : T \mid \Delta \qquad T \leq T'}{\Gamma \vdash M : T' \mid \Delta}$$

Each index (i or j) ranges over a finite set. The notation $[\cdots]_i$ denotes a finite sequence of judgments. In each rule, the same indices are understood to range over the same set. For example, in the third rule, i ranges over a finite set I and j over a finite set $J(i)$ depending on i, and these I and $J(i)$ are shared between the assumptions.

1.3. **Type system of the target calculus.** The characterization of strong termination is verified through the type theory of the target calculus we present here. The type theory is based on the standard intersection type discipline. We assume a special atomic type $\perp\!\!\!\perp$ and

write $\neg(\text{-})$ in place of $(\text{-}) \to \perp\!\!\!\perp$. We define the strict types τ, κ, and σ and types $\underline{\kappa}$ and $\underline{\sigma}$ by the following:

$$\begin{aligned} \sigma &::= \alpha \mid \underline{\sigma} \to \tau & \underline{\sigma} &::= \bigcap \sigma \\ \kappa &::= \neg\underline{\sigma} & \underline{\kappa} &::= \bigcap \kappa \\ \tau &::= \neg\underline{\kappa} \end{aligned}$$

where α represents atomic types. $\bigcap \kappa$ denotes a finite formal intersection $\kappa_1 \cap \kappa_2 \cap \cdots \cap \kappa_n$ with $n \geq 1$. $\bigcap \sigma$ is similar. Intersection follows associativity and commutativity. We have the subtype relation $\leq$ between types, which is defined naturally.

A typing judgment $\Pi, \Theta \vdash_s M : \rho$ has two environments, Π and Θ, the former a finite sequence of $x : \underline{\sigma}$ and the latter of $k : \underline{\kappa}$. Here M is either of term T, jump Q, value W, and continuation K, on which the kind of type ρ depends. A term has type τ, a jump $\perp\!\!\!\perp$, a value σ, and a continuation κ. We note that only strict types occur on the right hand of $\vdash_s$ (this is indicated by the subscript).

The inference rules of the intersection type discipline are standard [vBa92], except that the sorts must be respected. For instance, the derivation rule for $W_1 W_2$ is

$$\frac{\Pi, \Theta \vdash_s W_1 : \underline{\sigma} \to \tau \qquad [\Pi, \Theta \vdash_s W_2 : \sigma_i]_i}{\Pi, \Theta \vdash_s W_1 W_2 : \tau}$$

Moreover, we include the inheritance rule for each sort; e.g.,

$$\frac{\Pi, \Theta \vdash_s T : \tau \qquad \tau \leq \tau'}{\Pi, \Theta \vdash_s T : \tau'}$$

to deal with η-rules. We refer the reader to [Has15] for the presentation of the complete set.

Theorem 1.6. *Let M be a term of the CCV $\lambda\mu$-calculus. If $\Pi,\ \Theta \vdash_s [\![M]\!] : \tau$ is derivable in the target calculus, then $\Gamma \vdash M : T \mid \Delta$ is derivable for some Γ, Δ, and T in the CCV $\lambda\mu$-calculus.*

Proof. The theorem is verified in [Has15] using the inverse translation. By inspection of the proof therein, we see that the induced derivation of $\Gamma \vdash M : T \mid \Delta$ contains neither ω nor $\mathfrak{m}$, provided the derivation of $\Pi,\ \Theta \vdash_s [\![M]\!] : \tau$ contains ω nowhere. □

2. Strong Normalization

The main theorems of this section are the following: (i) a CCV $\lambda\mu$-term M is strongly normalizable if and only if $[\![M]\!]$ is strongly normalizable (Thm. 2.31), and (ii) M is strongly normalizable if and only if M is typeable (Thm. 2.30). We note that the strong normalizability of lambda terms is not closed under $\beta\eta$-equality. It happens that $M =_{\beta\eta} N$ and M is strongly normalizable, whereas N is not. Hence the main result (i) is sensitive to the choice of the CPS translation. The following argument assumes the colon translation as defined in Preliminaries. The case of the standard CPS translation is briefly discussed in 2.33.

2.1. **Termination of $\beta_\mu\beta_{jmp}\eta_\mu$ reduction sequences.** We verify the termination of all reduction sequences consisting exclusively of β_μ, β_{jmp}, and η_μ. The reasons we prove it independently are the following: (a) it is used later in Prop. 2.13, (b) it holds regardless of types, (c) in the proof, we introduce the notion of places, which are used in the sequel, and (d) it explains why we regard the exchange of the μ-operator and let-operator as a reduction rule rather than an equality rule in our previous work.

Since we have equality axioms between terms, the notion of subterms is obscure. We substitute this concept with the novel notion of places. We assume to be given an infinite set of place symbols.

Definition 2.1. Suppose that a CCV $\lambda\mu$-term M_0 is given. A term M occurs at *place* p if $p \mathbin{@} M$ is derived by the following recursive process. Let p_0 be an arbitrary place symbol. We start with $p_0 \mathbin{@} M_0$, and apply the following operations recursively until we reach variables:

(1) If either $p \mathbin{@} (MN)$ or $p \mathbin{@} (M \restriction x := N)$, then $p \mathbin{@} M$ and $q \mathbin{@} N$ concurrently, where q is a fresh place symbol.
(2) If either $p \mathbin{@} (\lambda x.\, M)$ or $p \mathbin{@} (\mu k.\, [l]M)$, then $q \mathbin{@} M$, where q is a fresh place symbol.

We say that a place q occurs in M whenever $q \mathbin{@} N$ is derived for some N from $p \mathbin{@} M$ during the process (the case $p = q$ is inclusive). We also say that $q \mathbin{@} N$ occurs in M.

Here we assume to read $[k](L \restriction x := M)$ for $[k]L \restriction x := M$. For the associativity of the let-construct, either bracketing yields the same set of places up to the renaming of place symbols. For instance, the following term has five places:

$$\underset{p_0}{\uparrow}(\mu k.\, \underset{p_1}{\uparrow}[l](\underset{p_2}{\uparrow}\lambda z.\, \underset{p_3}{\uparrow}x)y) \restriction y := \underset{p_4}{\uparrow}x.$$

Namely, a place marks the location from which a term starts. Note that one place may mark several terms. For example, place p_0 marks both of the whole term and $\mu k.\, [l](\lambda z.\, x)y$. Likewise, place p_1 marks $(\lambda z.\, x)y$ and $\lambda z.\, x$. Each place marks all possible subterms starting at the position.

In the let-binding $M \restriction x := N$, we regard that the scope M is viewable from the bound term N. We extend the notion to future let-binding. We consider M to be visible from N in term $M \restriction x := \mu k.\, \cdots [k]N$ though it is not currently the scope of N, as a one-step β_μ-reduction yields $\mu k.\, \cdots [k]M \restriction x := N$, in which M turns out to be the scope of N. This idea naturally leads to the following definition.

Definition 2.2. Let p be a place occurring in a given term M_0. The *vision* $V(p)$ is the set of places in M_0 recursively defined as follows:

(1) If $p \mathbin{@} M$ marks the bound term of $p_1 \mathbin{@} L \restriction x := M$, then $V(p)$ is defined by $\bigcup_q (\{q\} \cup V(q))$, where q ranges over the set of all places in L (p_1 inclusive).
(2) If $p \mathbin{@} M$ is preceded by a jumper, as in $p_1 \mathbin{@} \mu k.\, \cdots [k]M \cdots$, then $V(p)$ is defined by $V(p_1)$. Intuitively, place p is superposed over p_1 and the intermediate places between μk and the jumper $[k]$ are invisible. If k is not bound, we deal with $V(p)$ by the following third rule (thus $V(p) = \emptyset$).
(3) For all other cases, we set $V(p) = \emptyset$.

Remark 2.3.

(1) The places in $V(p)$ occur physically to the left of p. Hence $V(p)$ is defined by induction from left to right.

(2) By definition, visions are transitive. That is, if $r \in V(q)$ and $q \in V(p)$, then $r \in V(p)$.
(3) The definition of visions is irrelevant to the bracketing of let-constructs for associativity. Namely, if $p \mathbin{@} N$ occurs in $L \restriction x := M \restriction y := N$ with $y \notin L$, then either bracketing yields the same set $V(p)$ up to the renaming of place symbols.
(4) In contrast, the interchange law of μ and let is annoying. If we returned to the identification of $(\mu k.\, J) \restriction x := M$ and $\mu k.\, (J \restriction x := M)$, where $k \notin M$ as commented in Rem. 1.1, then we would have the following double vision problem. Suppose p is the place of M in $p_1 \mathbin{@} \mu k.\, [l]L \restriction x := M$, where $k \notin M$. If we understand it to be $p_1 \mathbin{@} (\mu k.\, [l]L) \restriction x := M$, place p_1 is visible from p. On the other hand, if we regard the expression as $p_1 \mathbin{@} \mu k.\, [l](L \restriction x := M)$, the vision of p skips place p_1, jumping from $[l]$. Namely, the vision is affected by bracketing.

Let us write $q \prec p$ if q is immediately visible from p. In other words, $q \in V(p)$ holds, while $q \in V(r)$ and $r \in V(p)$ hold for no r.

Remark 2.4. The relation $q \prec p$ holds if and only if either of the following two cases happens. (i) $p \mathbin{@} N$ is the place of the bound term of $L \restriction x := N$, and q occurs in L. Furthermore, if q occurs in a let-expression $P \restriction y := Q$ inside L, q lies in the argument side, Q. (ii) $p \mathbin{@} N$ is the place preceded by a jumper, as in $p_1 \mathbin{@} (\mu l.\, \cdots [l]N \cdots)$ with $q \prec p_1$.

Definition 2.5. The *breadth* $|p|$ of a place p is defined by induction on a physical location from left to right. We define $|p|$ as the smallest natural number n satisfying $|q| < n$ for all places $q \in V(p)$. In particular, $|p| = 0$ if $V(p) = \emptyset$.

In other words, $|p|$ is the height of the tree of all sequences of places $q \prec q' \prec \cdots \prec p$ having p as its root.

The following is an example of visions and breadths. Let us consider

$$\underset{p_0 p_1}{uv} \restriction u := \underset{p_2}{(\mu l.} \underset{p_3}{[l]v)} \restriction v := \underset{p_4}{\mu k.} \underset{p_5}{[k]} (\lambda x. \underset{p_6}{\mu m.} \underset{p_7}{[k]x}).$$

Then $V(p_0) = V(p_1) = V(p_6) = \emptyset$ and $V(p_2) = V(p_3) = \{p_0, p_1\}$, while $V(p_4) = V(p_5) = V(p_7) = \{p_0, p_1, p_2, p_3\}$. Hence $|p_0| = |p_1| = |p_6| = 0$ and $|p_2| = |p_3| = 1$, while $|p_4| = |p_5| = |p_7| = 2$. Let us suppose that a one-step β_μ-reduction is applied to this term. There are three possibilities, one for μl and two for μk by the choices of bracketing. One of the two for μk inhales $(\mu l.[l]v) \restriction v := \Box$, yielding

$$\underset{\bar{p}_0 \bar{p}_1}{uv} \restriction u := \underset{\bar{p}_4}{\mu k.} \underset{\bar{p}'_2}{[k]} (\underset{\bar{p}'_3}{\mu l.[l]v}) \restriction v := (\underset{\bar{p}_5}{\lambda x.} \underset{\bar{p}_6}{\mu m.} \underset{\bar{p}''_2}{[k]} (\underset{\bar{p}''_3}{\mu l.[l]v}) \restriction v := \underset{\bar{p}_7}{x})$$

where we write bars over places for distinction. The numbers associated with the places are given so that the correspondences are clear. For example, $\bar{p}'_3$ and $\bar{p}''_3$ marking v in the copied terms correspond to p_3 in front of v before the reduction. For instance, we have $V(\bar{p}''_3) = \{\bar{p}_0, \bar{p}_1\}$ and $V(\bar{p}_5) = \{\bar{p}_0, \bar{p}_1, \bar{p}'_2, \bar{p}'_3\}$, while $V(\bar{p}_7) = \{\bar{p}_0, \bar{p}_1, \bar{p}''_2, \bar{p}''_3\}$. Note that the visions are essentially unaffected by the reduction. This is because the reduction simply shifts the copies of $\mu l.[l]v$ to the points, where jumpers $[k]$ point to the original location skipping the intermediate parts. The only exception is the place in front of μk. We have $V(\bar{p}_4) = \{\bar{p}_0, \bar{p}_1\}$, thus $|\bar{p}_4| = 1$, which decrease from $|p_4| = 2$.

We consider the general case of a one-step β_μ reduction. Let us suppose that the places are specified as follows.

$$\begin{array}{ccccc} M \upharpoonright x{:=}\mu k.\cdots[k]N\cdots & & \longrightarrow & & \mu k.\cdots[k]M \upharpoonright x{:=}N\cdots \\ \uparrow_{s} \quad \uparrow_{r_0} \quad \uparrow_{t} & & & & \uparrow_{\bar{r}_0} \quad \uparrow_{\bar{s}^{(i)}} \quad \uparrow_{\bar{t}} \end{array}$$

where the displayed jumper denotes the i-th occurrence of $[k]$ under an appropriate enumeration, $i = 1, 2, \ldots, n$. For each place after the reduction, there is a unique corresponding place p before it. There are n places $\bar{p}^{(i)}$ $(i = 1, 2, \ldots, n)$, one in each copy of M, which correspond to p in M. Namely, we have an n-to-one correspondence if we look from the term after the reduction. For the places $\bar{p}$ occurring elsewhere, there is a one-to-one correspondence to the places p before the reduction.

We let the notation $C[M]$ mean a term wherein $p \mathbin{@} M$ occurs for some p.

Lemma 2.6. *Let us consider the β_μ reduction $C[M \upharpoonright x := \mu k.\, J] \to C[\mu k.\, J']$, where $J' = J\{[k]\Box \mapsto [k]M \upharpoonright x := \Box\}$. We set the place symbols as above.*

(1) *The strict inequality $|\bar{r}_0| < |r_0|$ holds.*
(2) *$|\bar{p}| \le |p|$ holds for every place $\bar{p}$ occurring in $C[\mu k.\, J']$, where p denotes the place in $C[M \upharpoonright x := \mu k.\, J]$ associated with $\bar{p}$ by the (n-to-one and one-to-one) correspondence explained immediately above (read $\bar{p} = \bar{p}^{(i)}$ for some i if it occurs in a copy of M).*

Proof. (1) is evident since $s \in V(r_0)$. We note that $V(s)$ and $V(\bar{r}_0)$ has a one-to-one correspondence, and thus $|s| = |\bar{r}_0|$.

(2) Recall that the breadth equals the height of the tree of places ordered by $\prec$. It suffices to show that $\bar{q} \prec \bar{p}$ implies $q \prec p$, excluding the case $\bar{p} = \bar{r}_0$. (i) The case that $\bar{p}$ occurs in one of the copies of M. If $\bar{p} = \bar{s}^{(i)}$ in front of a copy of M, then $\bar{q} \prec \bar{r}_0$ holds, as $\bar{p}$ is superposed over $\bar{r}_0$. By the one-to-one correspondence between $V(\bar{r}_0)$ and $V(s)$, we have $q \prec s$. Otherwise, $\bar{p}$ occurs strictly inside one of the copies of M (namely, it is $\bar{p}^{(i)}$ for some i). Then, $\bar{q}$ stays inside the same copy of M, or to the left of $\bar{r}_0$ when $\bar{p}$ is preceded by a jumper. Namely, $\bar{q}$ never lies in the dotted part between $\bar{r}_0$ and $\bar{s}^{(i)}$. Hence $q \prec p$ holds from the beginning. (ii) The case that $\bar{p}$ occurs in none of the copies of M. If $\bar{p} = \bar{t}$, then $\bar{q}$ occurs in M. Now $q \prec r_0$ holds. Therefore, $q \prec t$ as t superposes over r_0. If $\bar{p} \neq \bar{t}$, then $\bar{q}$ does not lie in M. This follows from Rem. 2.4, since M is not a let-argument and none of the jumpers to the right of $\bar{t}$ points into M, as the scope of μ-binding in M must stay inside M. As both $\bar{q}$ and $\bar{p}$ stay outside M, places q and p are not affected by the reduction. Hence $q \prec p$ from the beginning. □

Next, we consider the β_{jmp}-reduction $C[[l]\mu k.\, J] \to C[J\{l/k\}]$. Each place $\bar{p}$ in $C[J\{l/k\}]$ occurs either in C or in $J\{l/k\}$. Hence, we can naturally associate a place p in $C[[l]\mu k.\, J]$ that locates in C or in J.

Lemma 2.7. *Let us consider the β_{jmp}-reduction $C[[l]\mu k.\, J] \to C[J\{l/k\}]$. With each place $\bar{p}$ in $C[J\{l/k\}]$ is associated p in $C[[l]\mu k.\, J]$, as explained immediately above. Then $|\bar{p}| = |p|$ holds.*

Proof. A crucial case is when the place p is preceded by a jumper $[k]$ in J. Then, $\bar{p}$ is preceded by $[l]$. If l is bound and $\bar{r}_0$ is the place of $\mu l \cdots$ occurring in the context C, then $|\bar{p}| = |\bar{r}_0|$. On the other hand, if we let r_1 denote the place of $\mu k.\, J$, we have $|p| = |r_1| = |r_0|$. Evidently, $|\bar{r}_0| = |r_0|$, so $|\bar{p}| = |p|$. If l is not bound, we have $|\bar{p}| = 0 = |p|$. □

Next, we consider the η_μ-reduction $C[\mu k.\,[k]M] \to C[M]$. Each place $\bar{p}$ in $C[M]$ occurs either in C or in M. There is a corresponding place p in $C[\mu k.\,[k]M]$ located in C or in M.

Lemma 2.8. *Let us take η_μ-reduction $C[\mu k.\,[k]M] \to C[M]$. As explained above, a place p in $C[\mu k.\,[k]M]$ is associated with each place $\bar{p}$ in $C[M]$ Then $|\bar{p}| = |p|$ holds.*

Proof. Let $\bar{r}_1$ denote the place of M after reduction. Moreover, let r_0 denote the place of $\mu k.\,[k]M$. Then, $|\bar{r}_1| = |r_0| = |r_1|$. For other places, the lemma is immediate. □

We call p a *μ-place* if $p \mathbin{@} \mu k.\,J$ happens.

Definition 2.9. The *sight* of a term M is the natural sum of $\omega^{|p|}$, where p ranges over all μ-places occurring in M.

Written in a Cantor normal form, the sight of M is equal to $\omega^{n_1}k_1 + \omega^{n_2}k_2 + \cdots + \omega^{n_s}k_s$ for integers $k_i > 0$ and $n_1 > n_2 > \cdots > n_s \geq 0$, where k_i is the number of μ-places p satisfying $|p| = n_i$.

Proposition 2.10. *If $M_0 \to M_1$ by an application of rule β_μ, β_{jmp}, or η_μ, the sight of M_1 is strictly less than the sight of M_0.*

Proof. For the β_μ-reduction, we follow the notation in Lem. 2.6. By (2) of the lemma, thhe breadth of each place never increases. Each μ-place p in M has n copies after reduction. However, we have $|p| < |r_0|$ since $p \in V(r_0)$. To summarize, the places of breadth less than $|r_0|$ may be copied, while the places of breadth greater than or equal to $|r_0|$ are never copied. Moreover, the breadth of r_0 itself decreases by (1) of the same lemma. Hence the sight diminishes.

For the β_{jmp}-reduction, we use symbols in the proof of Lem. 2.7. The breadth of p never changes unless $p = r_1$. Moreover, the place r_1 just vanishes. For the η_μ-reduction, we use Lem. 2.8. The place r_0 vanishes. □

Corollary 2.11. *All $\beta_\mu\beta_{jmp}\eta_\mu$-reduction sequences are finite.*

Proof. Transfinite induction up to ω^ω by Prop. 2.10. □

Remark 2.12. To understand the necessity of regarding the exchange of the μ-operator and let-operator as reduction, we consider the following example:

$$K \restriction x := L \restriction y := \mu k.\,[m]M \restriction z := \mu l.\,[k]N.$$

We assume k does not occur elsewhere. By applying a β_μ-reduction to μk, we obtain $\mu k.\,[m]M \restriction z := \mu l.\,[k](K \restriction x := L \restriction y := N)$. Suppose $l \notin N$. If we can exchange μ and let, the term is equal to

$$\mu k.\,[m]M \restriction z := (\mu l.\,[k]K) \restriction x := L \restriction y := N.$$

Now M turns out to be visible from L, while it was previously out of vision, as M occurred to the right of L. Hence the vision of L is widened by the reduction. Furthermore, M becomes visible from N, while it was previously skipped by jumper $[k]$. These phenomena make the proof of Lem. 2.6 fail.

This situation is troublesome since the problem does not arise if $l \in N$. In this case, we cannot narrow the scope of μk. The behavior is influenced by whether $l \in N$ or not. We would need an argument sensitive to the occurrences of variables.

2.2. **Characterization of strong normalizability.** We characterize strong normalizable terms in the CCV $\lambda\mu$-calculus by the union-intersection type discipline and by the CPS translation. Strong normalizability is not closed under $\beta\eta$-convertibility. So we must be sensitive to the choice of the CPS. We assume the CPS defined via colon translation. At the end of this section, we sketch how to generalize the results to the standard translation

Proposition 2.13. *Let M be a term of the CCV $\lambda\mu$-calculus. If M is strongly normalizable, $[\![M]\!]$ is strongly normalizable.*

Proof. Toward contradiction, we assume that M is strongly normalizable, while $[\![M]\!]$ admits an infinite reduction sequence. By Lem. 1.3, we have an infinite sequence of practical reductions from $[\![M]\!]^{-1}$ with no use of the rule *exch*. By Cor. 2.11, we cannot continue β_μ, β_{jmp}, and η_μ ceaselessly. Hence there must be an infinite number of reductions other than these four rules. By Lem. 1.4 and 1.2, an infinite reduction sequence from $M^\dagger$ is thus induced. This contradicts the strong normalizability of M as $M \xrightarrow{*} M^\dagger$. □

Lemma 2.14. *In the target calculus, if T is strongly normalizable, then there is a derivation tree of $\Pi,\ \Theta\ \vdash_s\ T : \tau$ for some Π, Θ, and τ. Similar results hold for other sorts.*

Proof. This lemma is essentially the same as Cor. 3.4.4 of [vBa92, p. 159]. We note that, in the ordinary lambda calculus, strong normalizability with respect to $\beta\eta$ obviously implies strong normalizability with respect to β. As it can be easily checked, eacn term in β-normal form admits a type. Then, by induction on the length of the longest reduction sequences, we can verify the lemma. □

Proposition 2.15. *Let M be a CCV $\lambda\mu$-term. If M is strongly normalizable, there is a derivation tree of a typing judgment $\Gamma\ \vdash\ M : T \mid \Delta$ for some Γ, Δ, and T.*

Proof. By Prop. 2.13, $[\![M]\!]$ is strongly normalizable. Hence, by Lem. 2.14, we obtain a derivation tree $\Pi,\ \Theta\ \vdash_s\ [\![M]\!] : \tau$. By Thm. 1.6, we have $\Gamma\ \vdash\ M : T \mid \Delta$. □

The main contribution of this section is the verification of the inverse of Prop. 2.13 and 2.15. To this end, we elaborate a syntactic translation that satisfies a kind of soundness with respect to reduction. In the literature, we can find several proofs of strong normalizability by syntactic translations for call-by-value calculi with control operators [Nak03][IN06][KA08]. Though all use CPS translations, they have to manage the phenomenon that continuations may be discarded. All of these works add some twists to the translations to avoid this problem. Nakazawa preprocesses certain terms in the source calculus before the translation, carefully specifying harmful parts in the translation. Ikeda and Nakazawa modify the CPS translation by adjoining extra terms they call garbage. Kameyama and Asai use a two-level λ-calculus as the target of the translation to isolate the segment that does not discard continuations. See Rem. 2.19 for more information.

Our translation adds garbage, inspired by [IN06]. In order to deal with the complication causeed by associative let-binding, we add a new infix binary operator $(\text{-}) \cdot (\text{-})$ to the ordinary lambda calculus. We assume that the operator is syntactically associative:

$$(L \cdot M) \cdot N \ = \ L \cdot (M \cdot N).$$

We write $L \cdot M \cdot N$ for either of the two bracketings. We consider the standard $\beta\eta$-reduction. As a new reduction rule involving the dot operator, we add

$$M \cdot N \quad \to \quad N.$$

We define a new type of colon translation. The target calculus is the lambda calculus augmented by the binary dot operator.

Definition 2.16. We associate a fresh variable $\tilde{k}$ with each of continuation variables k in a one-to-one manner. We define $\tilde{K}$ as follows:

$$\tilde{K} = \begin{cases} \tilde{k} & \text{if } K = k \\ Q & \text{if } K = \lambda x.\, Q \end{cases}.$$

Each occurrence of variable x becomes free in the second case. We emphasize that $\tilde{k}$ is a variable independent from k. So the substitution $k \mapsto K$ does not automatically substitute $\tilde{k}$ with $\tilde{K}$.

Definition 2.17. Let M be a term in the CCV $\lambda\mu$-calculus, and let K be a lambda term. The lambda terms $\{\!\{M\}\!\}[K], \{\!\{J\}\!\}$, and V^* are simultaneously defined as in the following table:

$$\begin{array}{lcl}
\{\!\{V\}\!\}[K] & = & KV^* \\
\{\!\{L \upharpoonright x := M\}\!\}[K] & = & \{\!\{L\}\!\}[K] \cdot \{\!\{M\}\!\}[\lambda x.\, \tilde{K} \cdot \{\!\{L\}\!\}[K]] \\
\{\!\{V_1 V_2\}\!\}[K] & = & V_1^* K \tilde{K} V_2^* \\
\{\!\{VN\}\!\}[K] & = & \tilde{K} \cdot \{\!\{Vz \upharpoonright z := N\}\!\}[K] \\
\{\!\{NM\}\!\}[K] & = & \tilde{K} \cdot \{\!\{zM \upharpoonright z := N\}\!\}[K] \\
\{\!\{\mu k.\, J\}\!\}[K] & = & \tilde{K} \cdot \{\!\{J\}\!\}\{K/k,\, \tilde{K}/\tilde{k}\} \\
\{\!\{[k]M\}\!\} & = & \tilde{k} \cdot \{\!\{M\}\!\}[k] \\
\{\!\{J \upharpoonright x := M\}\!\} & = & \{\!\{J\}\!\} \cdot \{\!\{M\}\!\}[\lambda x.\, \{\!\{J\}\!\}] \\
x^* & = & x \\
(\lambda x.\, M)^* & = & \lambda k \tilde{k} x.\, \{\!\{M\}\!\}[k] \cdot ((\lambda x.\, \tilde{k} \cdot \{\!\{M\}\!\}[k])x)
\end{array}$$

where V denotes a value and N a non-value. L and M are understood to be arbitrary terms, while z is a fresh variable. We assume the infix dot operator has higher precedence than lambda binding. Though the definition is not a simple induction on construction, well-definedness is easy.

The obtained lambda terms do not follow the rules of sorts given in §1.1. We may renew the definition to accommodate the new colon translation. The sorts, however, scarcely play a role hereafter, as inverse translation will not involved. We use symbols such as K and Q only to indicate correspondence to the original target calculus.

Remark 2.18. As easily seen from Def. 2.17, the associated $\tilde{K}$ actually occurs in $\{\!\{M\}\!\}[K]$, although K may vanish.

Remark 2.19. We compare Def. 2.17 with the colon translation given in Preliminaries. A crucial difference is that K is actually substituted in the definition of the translation of $\mu k.\, J$. This type of translations is found in the verification of strong normalizability [Nak03][IN06]. In fact, the first attempt by Parigot for the call-by-name $\lambda\mu$-calculus already used a translation where the continuation was actually substituted [Par97] (unfortunately, the proof has a flaw; see [NT03]). If k does not occur in J, the substituted K vanishes. This is why we add the prefix $\tilde{K}$. That is, we record the history of the continuations K, that may be deleted. For a technical reason, the order of a value and a continuation is reversed in the translation of $V_1 V_2$, and the garbage $\tilde{K}$ is added. The complication of $(\lambda x.\, M)^*$ is nothing more than for proof to work out.

Lemma 2.20. *Let M_1 and M_2 be CCV $\lambda\mu$-terms. If $M_1 = M_2$ holds, then $\{\!|M_1|\!\}[K] = \{\!|M_2|\!\}[K]$ holds for every K. Likewise, if $J_1 = J_2$, then $\{\!|J_1|\!\} = \{\!|J_2|\!\}$ holds.*

Proof. First, both $\{\!|L \upharpoonright x := (M \upharpoonright y := N)|\!\}[K]$ and $\{\!|(L \upharpoonright x := M) \upharpoonright y := N|\!\}[K]$ are equal to $Q' \cdot Q'' \cdot \{\!|N|\!\}[\lambda y.\, \tilde{K} \cdot Q' \cdot Q'']$ where we set $Q' = \{\!|L|\!\}[K]$ and $Q'' = \{\!|M|\!\}[\lambda x.\, \tilde{K} \cdot Q']$. Second, both $\{\!|[k](L \upharpoonright x := M)|\!\}$ and $\{\!|([k]L) \upharpoonright x := M|\!\}$ are equal to $\tilde{k} \cdot Q' \cdot \{\!|M|\!\}[\lambda x.\, \tilde{k} \cdot Q']$ for $Q' = \{\!|L|\!\}[k]$. We comment that, in these two cases, the associativity of the infix dot operator is indispensable. Since the definition in 2.17 is compositional, the lemma follows. □

Remark 2.21. For a fresh variable k, the equality $(\{\!|M|\!\}[k])\{K/k,\, \tilde{K}/\tilde{k}\} = \{\!|M|\!\}[K]$ holds as naturally supposed. We note that the definition of $\{\!|M|\!\}[k]$ contains $\tilde{k}$ implicitly. Therefore, substituting only k with K does not suffice.

Lemma 2.22. *Let us put $\{\!|M|\!\}[K] = Q$ where $x \notin K$, and let us consider two substitutions, $\theta_0 = \{k \mapsto K,\, \tilde{k} \mapsto \tilde{K}\}$ and $\theta = \{k \mapsto \lambda x.\, \tilde{K} \cdot Q,\, \tilde{k} \mapsto \tilde{K} \cdot Q\}$. Then $\{\!|J|\!\}\theta = \{\!|J\{[k]\Box \mapsto [k]M \upharpoonright x := \Box\}|\!\}\theta_0$ holds.*

Proof. The proof is by induction on the construction of terms and jumps. We simultaneously verify $V^*\theta = (V\{[k]\Box \mapsto [k]M \upharpoonright x{:=}\Box\})^*\theta_0$ and $\{\!|L|\!\}\theta[K'] = \{\!|L\{[k]\Box \mapsto [k]M \upharpoonright x{:=}\Box\}|\!\}\theta_0[K']$. Here the tricky notation $\{\!|L|\!\}\theta[K']$ means $\{\!|L|\!\}[K']\theta$ under the condition that $k, \tilde{k} \notin K'$. During the induction process, however, the variables k and $\tilde{k}$ may occur in the position of K'. In this case, $\{\!|L|\!\}[K']\theta$ should be manipulated as $\{\!|L|\!\}\theta[K'\theta]$. An essential case is $J = [k]L$. The left hand side is $\{\!|J|\!\}\theta = \tilde{K} \cdot Q \cdot \{\!|L|\!\}\theta[\lambda x.\, \tilde{K} \cdot Q]$ by Rem. 2.21, while the right hand side equals $\tilde{K} \cdot Q \cdot \{\!|L\{[k]\Box \mapsto [k]M \upharpoonright x := \Box\}|\!\}\theta_0[\lambda x.\, \tilde{K} \cdot Q]$. Apply the induction hypothesis to L. □

Lemma 2.23. *Suppose $M_0 \to M_1$, where the whole M_0 is a redex that is contracted. Then $\{\!|M_0|\!\}[K] \xrightarrow{+} \{\!|M_1|\!\}[K]$ by one or more steps of $\beta\eta$ reduction (read $\{\!|J_0|\!\} \xrightarrow{+} \{\!|J_1|\!\}$ for the case of β_{jmp}). Moreover, $(\lambda x.\, Vx)^* \xrightarrow{+} V^*$ holds.*

Proof. First, consider the rule *exch*. We have $\{\!|(\mu k.\, J) \upharpoonright x{:=}M|\!\}[K] = \tilde{K} \cdot Q' \cdot \{\!|M|\!\}[\lambda x.\, \tilde{K} \cdot \tilde{K} \cdot Q']$, where $Q' = \{\!|J|\!\}\{K/k,\, \tilde{K}/\tilde{k}\}$. On the other hand, $\{\!|\mu k.\, (J \upharpoonright x := M)|\!\}[K]$ is equal to $\tilde{K} \cdot Q' \cdot \{\!|M|\!\}[\lambda x.\, Q']$. Therefore, the elimination of the two occurrences of $\tilde{K}$ settles this case. We comment that $\lambda x.\, \tilde{K} \cdot \tilde{K} \cdot Q'$ actually occurs, in view of Rem. 2.18. Hence a positive number of β reductions is enforced by the elimination.

For rule ad_1, we have $\{\!|NM|\!\}[K] = \tilde{K} \cdot \{\!|zM \upharpoonright z := N|\!\}[K]$, regardless of whether M is a value or a non-value. Hence, eliminating $\tilde{K}$ yields $\{\!|zM \upharpoonright z := N|\!\}[K]$. Rule ad_2 is similarly handled. For rule β_λ, we have $\{\!|(\lambda x.\, M)V|\!\}[K] = (\lambda k\tilde{k}x.\, (\{\!|M|\!\}[k] \cdot ((\lambda x.\, \tilde{k} \cdot \{\!|M|\!\}[k])x)))K\tilde{K}V^*$. Now, applying β reduction to $k, \tilde{k}$, and x yields $\{\!|M|\!\}[K] \cdot (\lambda x.\, \tilde{K} \cdot \{\!|M|\!\}[K])V^*$, which is equal to $\{\!|M \upharpoonright x := V|\!\}[K]$. This finishes the case of β_λ. Furthermore, from the last term, eliminating $\{\!|M|\!\}[K]$ and $\tilde{K}$ and applying a β reduction to x yield $\{\!|M|\!\}[K]\{V^*/x\}$. It is safe to assume that K and $\tilde{K}$ contain no x, by α-conversion if needed. Therefore the last terms equals $\{\!|M\{V/x\}|\!\}[K]$. This completes the case of rule β_{let}.

For rule β_μ, we have $\{\!|M \upharpoonright x := \mu k.\, J|\!\}[K] = Q \cdot \tilde{K} \cdot Q \cdot \{\!|J|\!\}\theta$ where Q and θ are given in Lem. 2.22. Now the elimination of the two Q's gives $\tilde{K} \cdot \{\!|J|\!\}\theta$. It is equal to $\{\!|\mu k.\, J\{[k]\Box \mapsto [k]M \upharpoonright x := \Box\}|\!\}[K]$ by the same lemma, finishing this case. We comment that $Q = \{\!|M|\!\}[K]$ is annihilated exactly at the moment of the dispatch of continuation $M \upharpoonright x := \Box$. This ensures reductions antecedently done inside M are not ignored even if

$k \notin J$. For rule β_{jmp}, we have $\{\!|[l]\mu k.\, J|\!\} = \tilde{l} \cdot \tilde{l} \cdot \{\!|J|\!\}\{l/k, \tilde{l}/\tilde{k}\}$. Eliminating two $\tilde{l}$'s gives $\{\!|J|\!\}\{l/k, \tilde{l}/\tilde{k}\} = \{\!|J\{l/k\}|\!\}$.

For rule η_{let}, we have $\{\!|x \upharpoonright x := M|\!\}[K] = Kx \cdot \{\!|M|\!\}[\lambda x.\, \tilde{K} \cdot Kx]$. The elimination of Kx and $\tilde{K}$ yields $\{\!|M|\!\}[\lambda x.\, Kx]$. Now we apply the η reduction to obtain $\{\!|M|\!\}[K]$. For rule η_λ, we must verify $(\lambda x.\, Vx)^* \xrightarrow{+} V^*$. The left hand side equals $\lambda k\tilde{k}x.\, (V^*k\tilde{k}x) \cdot ((\lambda x.\, \tilde{k} \cdot V^*k\tilde{k}x)x)$. Eliminating $V^*k\tilde{k}x$ and $\tilde{k}$ gives $\lambda k\tilde{k}x.\, (\lambda x.\, V^*k\tilde{k}x)x$. Hence, one step of β followed by three steps of η yields V^*. Thus $\{\!|\lambda x.\, Vx|\!\}[K] \xrightarrow{+} \{\!|V|\!\}[K]$ is also valid. Finally, if $k \notin M$, then $\{\!|\mu k.\, [k]M|\!\}[K] = \tilde{K} \cdot \tilde{K} \cdot \{\!|M|\!\}[K]$. Eliminating two $\tilde{K}$'s gives $\{\!|M|\!\}[K]$, establishing the case of the η_μ rule. □

Lemma 2.23 ensures the strict preservation of reductions in the case where the redex occurs naked at the topmost level. In general, the redex R may be encapsulated in a context as $C[R]$. We introduce the notion of E-depth to handle this.

We recall the definition of places in Def. 2.1 where the notion of an occurrence of $q \mathbin{@} N$ in M was also defined.

In the following definition, we use the *evaluation contexts* E defined by the following syntax:

$$E ::= \Box \mid E[V\Box] \mid E[\Box M] \mid E[M \upharpoonright x := \Box].$$

Definition 2.24. The *E-depth* $d_M(q \mathbin{@} L)$ is defined, when $q \mathbin{@} L$ occurs in M. As the base case, we set $d_L(q \mathbin{@} L) = 0$. In general, the definition is provided by the following table, where $q \mathbin{@} L$ is assumed to occur in M or V.

$$\begin{array}{lcl}
d_{E[M]}(q \mathbin{@} L) & = & d_M(q \mathbin{@} L) \\
d_{E[VN]}(q \mathbin{@} L) & = & 1 + d_V(q \mathbin{@} L) \\
d_{E[NM]}(q \mathbin{@} L) & = & 1 + d_M(q \mathbin{@} L) \\
d_{E[M \upharpoonright x:=N]}(q \mathbin{@} L) & = & 1 + d_M(q \mathbin{@} L) \\
d_{\lambda x.\, M}(q \mathbin{@} L) & = & 1 + d_M(q \mathbin{@} L) \\
d_{\mu k.\, [l]M}(q \mathbin{@} L) & = & 1 + d_M(q \mathbin{@} L).
\end{array}$$

The second through the fourth of these equalities handle the case where L occurs in the context E if we split the term to the shape $E[M]$. The number d_M changes by bracketing of let-binding, This does no harm, however, for we use the number only as the measure of complexity of terms to handle one-step reductions.

We prove theorems by induction on the E-depth. We first explore the base case, in which the redex R has E-depth 0. Namely, the redex occurs in the form $E[R]$. For rule β_{jmp}, we understand R to be $\mu m.\, [l]\mu k.\, J$, including the μ-operator preceding the redex $[l]\mu k.\, J$. We must beware of rule η_λ, the redex of which is a value.

Lemma 2.25. *Given an evaluation context E and a term K in the target calculus, there are Q_E and $\mathcal{K}_E$ that satisfy the following:*

(1) *If N is a non-value, $\{\!|E[N]|\!\}[K] = Q_E \cdot \{\!|N|\!\}[\mathcal{K}_E]$.*
(2) *If V is a value, $Q_E \cdot \{\!|V|\!\}[\mathcal{K}_E] \xrightarrow{*} \{\!|E[V]|\!\}[K]$.*

Here Q_E and $\mathcal{K}_E$ are terms in the extended language, although the former may be void. If this is the case, we just ignore the preceding Q_E and the following dot.

Proof. (1) The definition of Q_E and $\mathcal{K}_E$ will be read off from the following equalities:

$$\begin{aligned}
\{\!\{E[NM]\}\!\}[K] &= Q_E \cdot \tilde{\mathcal{K}}_E \cdot \{\!\{zM\}\!\}[\mathcal{K}_E] \cdot \{\!\{N\}\!\}[\lambda z.\, \tilde{\mathcal{K}}_E \cdot \{\!\{zM\}\!\}[\mathcal{K}_E]] \\
\{\!\{E[VN]\}\!\}[K] &= \ Q_E \cdot \tilde{\mathcal{K}}_E \cdot \{\!\{Vz\}\!\}[\mathcal{K}_E] \cdot \{\!\{N\}\!\}[\lambda z.\, \tilde{\mathcal{K}}_E \cdot \{\!\{Vz\}\!\}[\mathcal{K}_E]] \\
\{\!\{E[M \upharpoonright x := N]\}\!\}[K] &= \qquad Q_E \cdot \{\!\{M\}\!\}[\mathcal{K}_E] \cdot \{\!\{N\}\!\}[\lambda x.\, \tilde{\mathcal{K}}_E \cdot \{\!\{M\}\!\}[\mathcal{K}_E]]
\end{aligned}$$

Namely, the following inductive definition works. If $E = \Box$, we set $\mathcal{K}_\Box = K$ and $Q_\Box$ as void. For inductive cases,

$$\begin{aligned}
Q_{E[\Box M]} &= Q_E \cdot \tilde{\mathcal{K}}_E \cdot \{\!\{zM\}\!\}[\mathcal{K}_E] & \mathcal{K}_{E[\Box M]} &= \lambda z.\, \tilde{\mathcal{K}}_E \cdot \{\!\{zM\}\!\}[\mathcal{K}_E] \\
Q_{E[V\Box]} &= Q_E \cdot \tilde{\mathcal{K}}_E \cdot \{\!\{Vz\}\!\}[\mathcal{K}_E] & \mathcal{K}_{E[V\Box]} &= \lambda z.\, \tilde{\mathcal{K}}_E \cdot \{\!\{Vz\}\!\}[\mathcal{K}_E] \\
Q_{E[M\upharpoonright x:=\Box]} &= Q_E \cdot \{\!\{M\}\!\}[\mathcal{K}_E] & \mathcal{K}_{E[M\upharpoonright x:=\Box]} &= \lambda x.\, \tilde{\mathcal{K}}_E \cdot \{\!\{M\}\!\}[\mathcal{K}_E].
\end{aligned}$$

The first two in the left column depend on the choices of fresh variables z in the construction of $\{\!\{\cdot\}\!\}$.

(2) By induction on the construction of E. In the case $E = \Box$ and $E[M \upharpoonright x := \Box]$, both sides are equal. In the case $E[\Box M]$, by definition, $Q_{E[\Box M]} \cdot \{\!\{V\}\!\}[\mathcal{K}_{E[\Box M]}]$ reduces to $Q_E \cdot \{\!\{zM \upharpoonright z := V\}\!\}[\mathcal{K}_E]$ by dropping a single $\tilde{\mathcal{K}}_E$. By Lem. 2.23, it reduces to $Q_E \cdot \{\!\{VM\}\!\}[\mathcal{K}_E]$, viz., $\{\!\{E[VM]\}\!\}[K]$. The case of $E[V'\Box]$ is similar. □

Lemma 2.26. *Let $R \to S$ be one of the reduction rules. Then, $\{\!\{E[R]\}\!\}[K] \xrightarrow{+} \{\!\{E[S]\}\!\}[K]$ with one or more steps of $\beta\eta$ reduction.*

Proof. Except for the η_λ-redex, R is a non-value. By Lem. 2.25, (1), $\{\!\{E[R]\}\!\}[K] = Q_E \cdot \{\!\{R\}\!\}[\mathcal{K}_E]$, which contracts to $Q_E \cdot \{\!\{S\}\!\}[\mathcal{K}_E]$ by one or more steps by Lem. 2.23. If S is a non-value, we are done. If S is a value, we employ Lem. 2.25, (2). If R equals $\lambda x.\, Vx$, which is a value, we need to take special care. In the case of $E[\Box N]$, the translation $\{\!\{E[(\lambda x.\, Vx)N]\}\!\}[K]$ is as computed in the proof of Lem. 2.25. We apply $(\lambda x.\, Vx)^* \xrightarrow{+} V^*$, verified in Lem. 2.23, to the two occurrences of $\{\!\{(\lambda x.\, Vx)z\}\!\}[\mathcal{K}_E] = (\lambda x.\, Vx)^* \mathcal{K}_E \tilde{\mathcal{K}}_E z$. The other cases are similar. □

So the base case is done. Now, by induction on E-depth, we can verify a central proposition of this subsection.

Proposition 2.27. *If $L \xrightarrow{+} M$ holds in the CCV $\lambda\mu$-calculus, then $\{\!\{L\}\!\}[K] \xrightarrow{+} \{\!\{M\}\!\}[K]$ holds with respect to $\beta\eta$ reduction for every K.*

Proof. It suffices to prove the case of the one-step reduction $L \to L_1$. Let $R \to S$ be the instance of the reduction rule contracted by this step, and let $q \mathbin{@} R$ be the place of the redex in L. We show that if $d_L(q \mathbin{@} R) \leq m$, then $L \to L_1$ implies $\{\!\{L\}\!\}[K] \xrightarrow{+} \{\!\{L_1\}\!\}[K]$, and simultaneously that if $d_V(q \mathbin{@} R) \leq m$, then $V \to V_1$ implies $V^* \xrightarrow{+} V_1^*$ by induction on m. We note that if $V \to V_1$ and if V is a value, V_1 is also a value. The base case $m = 0$ is Lem. 2.26. We verify the induction step.

(i) First, we consider the case where $q \mathbin{@} R$ occurs in the V in $L = E[VM']$. We split cases further according to whether M' is a value. If it is a value W, we have $\{\!\{E[VW]\}\!\}[K] = \tilde{\mathcal{K}}_E \cdot V^* \mathcal{K}_E \tilde{\mathcal{K}}_E W^*$. Since $d_V(q \mathbin{@} R) < d_{E[VW]}(q \mathbin{@} R)$, we apply the induction hypothesis to V^*. If M' is a non-value N, the computation of $\{\!\{E[VN]\}\!\}[K]$ is displayed in Lem. 2.25. Note that $\{\!\{Vz\}\!\}[\mathcal{K}_E] = V^* \mathcal{K}_E \tilde{\mathcal{K}}_E z$. We apply the induction hypothesis to all occurrences of V^*. Observe that at least one occurrence of V^* exists.

(ii) The case where $q@R$ occurs in the M in $L = E[NM]$. The computation of $\{\!|E[NM]|\!\}[K]$ is given in Lem. 2.25. We note $d_{zM}(q \mathbin{@} R) = d_M(q \mathbin{@} R) < d_{E[NM]}(q \mathbin{@} R)$ since z is a value while N is a non-value. Apply the induction hypothesis to all occurrences of $\{\!|zM|\!\}[\mathcal{K}_E]$.

(iii) The case where $q \mathbin{@} R$ occurs in the M in $L = E[M \upharpoonright x := M']$. The computation of $\{\!|E[M \upharpoonright x := M']|\!\}[K]$ is given in Lem. 2.25, if we read $N = M'$. Since $d_M(q \mathbin{@} R) < d_{E[M \upharpoonright x:=M']}(q \mathbin{@} R)$, apply induction hypothesis to all occurrences of $\{\!|M|\!\}[\mathcal{K}_E]$.

(iv) Case where $q \mathbin{@} R$ occurs in M of $L = E[\lambda x.\, M]$. We have $\{\!|E[\lambda x.\, M]|\!\}[K] = Q_E \cdot \mathcal{K}_E[\lambda k\tilde{k}x.\, (\{\!|M|\!\}[k] \cdot ((\lambda x.\, \tilde{k} \cdot \{\!|M|\!\}[k])x))]$. Apply induction hypothesis to the two occurrences of $\{\!|M|\!\}[k]$ since $d_M(q \mathbin{@} R) < d_{E[\lambda x.\, M]}(q \mathbin{@} R)$. We note that, taking E to be void, this case essentially contains the proof of $V^* \xrightarrow{+} V_1^*$.

(v) Finally, we consider the case where $q \mathbin{@} R$ occurs in the M in $L = E[\mu k.\, [l]M]$. We have $\{\!|E[\mu k.\, [l]M]|\!\}[K] = Q_E \cdot \tilde{\mathcal{K}}_E \cdot \tilde{l} \cdot \{\!|M|\!\}[l]\{\mathcal{K}_E/k,\, \tilde{\mathcal{K}}_E/\tilde{k}\}$. Apply the induction hypothesis to $\{\!|M|\!\}[l]$.

□

We define the translation of the union-intersection types of CCV $\lambda\mu$-calculus into intersection types of the target calculus:

$$\alpha^* = \alpha, \quad (S \to T)^* = T^+ \to \perp\!\!\!\perp \to \neg S^*, \quad (\bigcap R)^* = \bigcap R^*$$

$$(\bigcup S)^+ = \bigcap \neg S^*$$

$$[\![T]\!] = \neg T^+$$

In the previous paper, we defined $(S \to T)^*$ as $S^* \to [\![T]\!]$. The modification corresponds to the change of the colon translation. Although the results of the type translation do not obey the rule of the target calculus in §1.3, this does not matter in the following argument. We simply ignore the distinction between σ, κ, and τ. Accordingly, there is no need to distinguish between the environments Π, Θ. We add the following inference rule:

$$\frac{\Pi \vdash_s Q_1 : \perp\!\!\!\perp \qquad \Pi \vdash_s Q_2 : \perp\!\!\!\perp}{\Pi \vdash_s Q_1 \cdot Q_2 : \perp\!\!\!\perp}$$

Lemma 2.28. *If* $\Gamma \vdash M : T \mid \Delta$ *is derived in the CCV* $\lambda\mu$*-calculus, then* $\Pi,\, k : T^+,\, \tilde{k} : \perp\!\!\!\perp \vdash_s \{\!|M|\!\}[k] : \perp\!\!\!\perp$ *is derived for some* Π.

Proof. This lemma was proved in our previous paper [Has15] for the original colon translation. We follow the same line. We need to pay attention, however, to the pieces added by the dot operator. Let us consider the case $\Gamma \vdash J \upharpoonright x := M : \perp\!\!\!\perp \mid \Delta$ inferred from $[\, \Gamma,\, x : S_i \vdash J : \perp\!\!\!\perp \mid \Delta\,]_i$ and $\Gamma \vdash M : \bigcup S_i \mid \Delta$. We have $\{\!|J \upharpoonright x := M|\!\}[k] = \{\!|J|\!\} \cdot \{\!|M|\!\}[\lambda x.\, \{\!|J|\!\}]$. We must take care of the free occurrences of x that may appear in the first $\{\!|J|\!\}$ in front of the dot. Hence, we choose i_0 and add $x : S_{i_0}^*$ to the typing environment. We then do the same for $L \upharpoonright x := M$. Next we consider the case $N_1 N_2$, where N_i are non-values. We have $\{\!|N_1 N_2|\!\}[k] = \tilde{k} \cdot \{\!|zN_2|\!\}[k] \cdot \{\!|N_1|\!\}[\lambda z.\, \tilde{k} \cdot \{\!|zN_2|\!\}[k]]$, where $\{\!|zN_2|\!\}[k] = \tilde{k} \cdot (zk\tilde{k}w) \cdot \{\!|N_2|\!\}[\lambda w.\, \tilde{k} \cdot (zk\tilde{k}w)]$. The variable z occurs freely in the left $\{\!|zN_2|\!\}[k]$, while the variable w occurs freely in the left $zk\tilde{k}w$ contained in $\{\!|zN_2|\!\}[k]$. We choose i_0 and add $z : \bigcap_j (T^+ \to \perp\!\!\!\perp \to \neg S_{i_0 j}^*)$ to the type environment Moreover, we choose $j_0(i)$ for each i and also add $w : \bigcap_i S_{i\, j_0(i)}^*$ to the type environment. Then, from the induction hypotheses on N_i, we can infer the typing of

$\{\!|N_1 N_2|\!\}[k]$ under the augmented environment, in spite of the free occurrences of z and w. The remaining cases are similar. □

Proposition 2.29. *If a CCV $\lambda\mu$-term M is typeable, then M is strongly normalizable.*

Proof. If M is typeable, $\{\!|M|\!\}[k]$ is typeable by Lem. 2.28. Then, as sketched in the Appendix, Cor. A.5, $\{\!|M|\!\}[k]$ is strongly normalizable in the lambda calculus with an additional rewriting rule for the dot operator. Therefore M is strongly normalizable by Prop. 2.27. □

Theorem 2.30. *A CCV $\lambda\mu$-term M is strongly normalizable if and only if M is typeable.*

Proof. A combination of Prop. 2.15 and 2.29. □

Theorem 2.31. *A CCV $\lambda\mu$-term M is strongly normalizable if and only if $[\![M]\!]$ is strongly normalizable.*

Proof. The only-if part is just Prop. 2.13. The converse uses the characterization by types. Provided that $[\![M]\!]$ is strongly normalizable, M is typeable. This is included in the proof of Prop. 2.15. Finally, M is strongly normalizable by Prop. 2.29. □

Remark 2.32. We call a CCV $\lambda\mu$-term strongly quasi-normalizable if all reduction sequences containing none of the η-type rules are finite. Let us show that strong quasi-normalizability implies strong normalizability. Suppose M is strongly quasi-normalizable. By inspection of the proof of Prop. 2.13, we see that $[\![M]\!]$ is strongly β-normalizable. Hence $[\![M]\!]$ is typeable with no use of ω. The rest of the proof goes as that of Thm. 2.31. Contrary to the case of weak normalizability [Has15], the inverse is trivial.

Remark 2.33. Let us briefly discuss how to extend the main theorem to the standard CPS translation. The translation is defined by the following:

$$
\begin{array}{lcl}
[\![V]\!] & := & \lambda k.\, kV^* \\
[\![MN]\!] & := & \lambda k.\, [\![M]\!](\lambda x.\, [\![N]\!](\lambda y.\, xyk)) \\
[\![M \upharpoonright x := N]\!] & := & \lambda k.\, [\![N]\!](\lambda x.\, [\![M]\!]k) \\
[\![\mu k.\, J]\!] & := & \lambda k.\, [\![J]\!] \\
[\![[k]M]\!] & := & [\![M]\!]k \\
[\![J \upharpoonright x := N]\!] & := & [\![N]\!](\lambda x.\, [\![J]\!]) \\
x^* & := & x \\
(\lambda x.\, M)^* & := & \lambda x.\, [\![M]\!].
\end{array}
$$

Theorem 2.31 remains valid for this definition. This is verified along the following line. For distinction, let $[\![M]\!]_c$ denote the CPS via the colon translation given in Preliminaries. Let us observe that, if we modify the standard CPS translation by $[\![\mu k.\, J]\!]' = \lambda k.\, (\lambda k.\, [\![J]\!]')k$ and by $(\lambda x.\, M)^* = \lambda xk.\, [\![M]\!]'k$, then we have $[\![M]\!]' \xrightarrow{*} [\![M]\!]_c$. Here $[\![M]\!]'$ is η-equal to $[\![M]\!]$. In the ordinary lambda calculus, strong normalizability is stable under η-equality (one way to verify this is to use the characterization by intersection types). Hence, if $[\![M]\!]$ is strongly normalizable, so are $[\![M]\!]'$ and, in turn, $[\![M]\!]_c$. Therefore M is strongly normalizable by the theorem 2.31. For the only-if part, if M is strongly normalizable, M is typeable by Prop. 2.15. We can prove that the typeability of M induces the typeability of $[\![M]\!]$ (verified for $[\![M]\!]_c$ in [Has15]). Hence $[\![M]\!]$ is strongly normalizable.

3. Conclusion

In the series of two papers, we presented call-by-value lambda calculi with control operators. They are complete with respect to the standard CPS semantics. The key idea is to introduce equality axioms between terms, departing from the convention that terms are freely generated by grammars.

We demonstrated the aptitude of the calculi through several mathematical properties. In this second paper, we gave the characterization of the strong termination property of the CCV $\lambda\mu$-calculus. We verified the following two results:

(1) M is strongly normalizing iff its CPS translation $[\![M]\!]$ is strongly normalizing.
(2) M is strongly normalizing iff M is typeable (with use of empty intersection or empty union nowhere).

We mention future problems that are not tackled in our series of papers. We adopted the reduction rule *exch* for the characterization of strong termination. It remains open if we assume the equality rule 1.1 exchanging μ and let in place. Second, we plan to extend the results to a $\lambda\mu$-calculus having delimited control operators. As a matter of fact, this work is a precursory extract from our attempt to develop complete calculi with delimited control operators.

In the literature, we can find the characterization of strong normalizability for various systems of the $\lambda\mu$-calculus[1]. Van Bakel, Barbanera, and de'Liguoro considered a call-by-name $\lambda\mu$-calculus using special forms of intersection and product types [VBD12]. Tsukada and Nakazawa introduced a polarized variation of $\bar{\lambda}\mu\tilde{\mu}$-calculus and gave a characterization using union and intersection types [TN16]. In particular, the latter may have a close connection to our results, although the details remain to be investigated in the future.

Appendix A. Strong normalizability of the extended lambda calculus

We give a sketch of the strong normalizability result needed in the proof of Prop. 2.29. We extend the ordinary lambda calculus by a binary dot operator $M \cdot N$, which is associative, i.e., $(L \cdot M) \cdot N = L \cdot (M \cdot N)$ holds. We omit the brackets. As a new reduction rule related to the dot operator, we add

$$M \cdot N \quad \to \quad N$$

in addition to the ordinary $\beta\eta$-reduction.

We introduce an intersection type system. We do not need sorts. So the strict types σ and types $\tau = \underline{\sigma}$ are defined simply by

$$\sigma ::= \alpha \mid \tau \to \sigma$$
$$\tau ::= \bigcap \sigma$$

where α ranges over atomic types and $\bigcap \sigma$ signifies a finite intersection $\sigma_1 \cap \sigma_2 \cap \cdots \cap \sigma_n$ ($n \geq 1$). We emphasize that the nullary intersection ω is not considered here. We assume that a special atomic type $\perp\!\!\!\perp$ is included.

We naturally define the subtype relation $\leq$. As typing rules, we add

$$\frac{\Gamma \ \vdash_s \ M : \perp\!\!\!\perp \qquad \Gamma \ \vdash_s \ N : \perp\!\!\!\perp}{\Gamma \ \vdash_s \ M \cdot N : \perp\!\!\!\perp}$$

[1]We thank an anonymous referee who informed us of the related works.

to the standard rules, including the inheritance rule. We verify the strong normalizability of this type system by the standard computability method [vBa92][MHH98].

Strong normalizability satisfies the following three properties elucidated in [MHH98], even if we consider η-reduction and the new rule associated with the dot operator: (i) if $L_1, L_2, \ldots, L_n$ $(n \geq 0)$ are strongly normalizable, $xL_1L_2\cdots L_n$ is strongly normalizable, (ii) if Mx is strongly normalizable, M is strongly normalizable. (iii) if N and $M\{N/y\}L_1L_2\cdots L_n$ $(n \geq 0)$ are strongly normalizable, $(\lambda y.\, M)NL_1L_2\cdots L_n$ is strongly normalizable.

Definition A.1. We define a family of sets Comp_τ of terms by induction on the construction of type τ.

$$M \in \mathit{Comp}_\alpha \iff M \text{ is strongly normalizable}$$
$$M \in \mathit{Comp}_{\tau\to\tau'} \iff \forall N \in \mathit{Comp}_\tau.\ MN \in \mathit{Comp}_{\tau'}$$
$$M \in \mathit{Comp}_{\tau\cap\tau'} \iff M \in \mathit{Comp}_\tau \cap \mathit{Comp}_{\tau'}.$$

It does no harm to consider non-typeable terms. So we do not include the condition of types for simplicity.

Lemma A.2. *The following hold for each type τ:*

(1) $x \in \mathit{Comp}_\tau$.
(2) *If $M \in \mathit{Comp}_\tau$, then M is strongly normalizable.*

Proof. Simultaneous induction on construction of types τ. To let induction go through, we strengthen condition (1): $xM_1M_2\cdots M_n \in \mathit{Comp}_\tau$ whenever $M_1, M_2, \ldots, M_n$ are strongly normalizable. To handle the case of $\tau \to \tau'$, we need property (i) given above. For (2) of $\tau \to \tau'$, we need (ii). We comment that the lemma fails if we allow the nullary intersection. □

Lemma A.3. *If $\tau \leq \tau'$ holds, $\mathit{Comp}_\tau \subseteq \mathit{Comp}_{\tau'}$ holds.* □

Lemma A.4. *Suppose that $x_1 : \tau_1,\ x_2 : \tau_2,\ \ldots,\ x_n : \tau_n \vdash_s M : \sigma$ holds. For every n-tuple of $P_i \in \mathit{Comp}_{\tau_i}$, we have $M\{P_1/x_1, P_2/x_2, \ldots, P_n/x_n\} \in \mathit{Comp}_\sigma$.*

Proof. Induction on the construction of the derivation trees of typing judgments. We consider the rule deriving $M \cdot N : \perp\!\!\!\perp$ from $M : \perp\!\!\!\perp$ and $N : \perp\!\!\!\perp$. The substitution P_i/x_i plays no role in this case, so we omit it. Since $\perp\!\!\!\perp$ is an atomic type, the induction hypotheses say M and N are strongly normalizable. Then, $M \cdot N$ is obiously also strongly normalizable, that is, $M \cdot N \in \mathit{Comp}_{\perp\!\!\!\perp}$. To handle lambda-abstraction, we need property (iii) given above. For the inheritance rule, we use Lem. A.3. □

Corollary A.5. *If $\Gamma \vdash_s M : \sigma$ is derivable, M is strongly normalizable.*

Proof. We take x_i as P_i in Lem. A.4. Then, noting (1) of Lem. A.2, we have $M \in \mathit{Comp}_\sigma$. Thus M is strongly normalizable by (2) of Lem. A.2. □

References

[Has15] R. Hasegawa, Complete call-by-value calculi with control operators, I, to appear.
[IN06] S. Ikeda and K. Nakazawa, Strong normalization proofs by CPS-translations, *Inform. Process. Let.* 99(4):163–170, 2006

[KA08] Y. Kameyama and K. Asai, Strong normalization of polymorphic calculus for delimited continuations, *Proceedings of Austrian-Japanese Workshop on Symbolic Computation in Software Science, SCSS 2008*, Hagenberg, Austria, Jul. 2008, RISC-Linz Report Series No. 08–08, pages 96–108.

[MHH98] J. Mitchell, M. Hoang, and B. T. Howard, Labeling techniques and typed fixed-point operators. *Higher Order Operational Techniques in Semantics*, A. D. Gordon, A. M. Pitts, eds., pages 137–174, Cambridge University Press, 1998.

[Mog88] E. Moggi, Computational lambda-calculus and monads, preprint, LFCS Report Series, Laboratory for Foundations of Computer Science, Department of Computer Science, The University of Edinburgh, Oct., 1988.

[Nak03] K. Nakazawa, Confluency and strong normalizability of call-by-value $\lambda\mu$-calculus, *Theoret. Comput. Sci.* 290(1):429–463, 2003.

[NT03] K. Nakazawa and M. Tatsuta, Strong normalization proof with CPS-translation for second order classical natural deduction, *J. Symbolic Logic* 68(3):851–859, 2003; *Corrigendum*, 68(4):1415–1416, 2003.

[Par97] M. Parigot, Proofs of strong normalisation for second order classical natural deduction, *J. Symbolic Logic* 62(4):1461–1479, 1997.

[Plo75] G. D. Plotkin, Call-by-name, call-by-value and the λ-calculus, *Theoret. Comput. Sci.* 1(2):125–159, 1975.

[TN16] T. Tsukada and K. Nakazawa, Intersection and union type assignment and polarised $\bar{\lambda}\mu\tilde{\mu}$, draft, 2016.

[vBa92] S. van Bakel, Complete restrictions of the intersection type discipline, *Theoret. Comput. Sci.*, 102(1):135–163, 1992.

[VBD12] S. van Bakel, F. Barbanera, and U. de'Liguoro, Characterisation of strongly normalising $\lambda\mu$-terms, Proceedings Sixth Workshop on Intersection Types and Related Systems, *Electr. Proc. Theoret. Comp. Sci.* 121:17–34, 2012.