INFORMATIQUE THÉORIQUE ET APPLICATIONS

KEN-ETSU FUJITA

Domain-free $\lambda\mu$-calculus

Informatique théorique et applications, tome 34, n° 6 (2000), p. 433-466

<http://www.numdam.org/item?id=ITA_2000__34_6_433_0>

Theoretical Informatics and Applications
Theoret. Informatics Appl. **34** (2000) 433–466

DOMAIN-FREE $\lambda\mu$-CALCULUS *

KEN-ETSU FUJITA[1]

Abstract. We introduce a domain-free $\lambda\mu$-calculus of call-by-value as a short-hand for the second order Church-style. Our motivation comes from the observation that in Curry-style polymorphic calculi, control operators such as `callcc` or μ-operators cannot, in general, handle correctly the terms placed on the control operator's left, so that the Curry-style system can fail to prove the subject reduction property. Following the continuation semantics, we also discuss the notion of values in classical system, and propose an extended form of values. It is proved that the CPS-translation is sound with respect to domain-free $\lambda 2$ (second-order λ-calculus). As a by-product, we obtain the strong normalization property for the second-order $\lambda\mu$-calculus of call-by-value in domain-free style. We also study the problems of type inference, typability, and type checking for the call-by-value system. Finally, we give a brief comparison with standard ML plus `callcc`, and discuss a natural way to avoid the unsoundness of ML with `callcc`.

Mathematics Subject Classification. 68N18, 68Q05.

INTRODUCTION

On the basis of the Curry–Howard–De Bruijn isomorphism [33], proof reductions can be regarded as computational rules, and the algorithmic contents of proofs can be used to obtain correct programs that satisfy logical specifications. The computational meaning of proofs has been investigated in a wide range of

Keywords and phrases: $\lambda\mu$-calculus, domain-free style, call-by-value, polymorphism, subject reduction, CPS-translation, strong normalization, Church–Rosser, type inference, type checking.

* *This is a revised and extended version of the paper entitled with Explicitly Typed $\lambda\mu$-Calculus for Polymorphism and Call-by-Value, presented at the 4th International Conference,* Typed Lambda Calculi and Applications *(TLCA '99), L'Aquila, Italy, April 7–9, 1999.*

[1] Shimane University, Department of Mathematics and Computer Science, Matsue 690-8504, Japan; e-mail: `fujiken@cis.shimane-u.ac.jp`

fields, including not only intuitionistic logic but also classical logic and modal logic [12, 13, 35, 37, 42]. In the area of classical logic, there have been a number of noteworthy investigations including Griffin [22], Murthy [43], Parigot [46], Berardi and Barbanera [4], Rehof and Sørensen [52], de Groote [24], Ong [44], and Ong and Stewart [45].

As far as we know, however, polymorphic call-by-value calculus has never been studied from the viewpoint of classical logic. In this paper, we introduce a domain-free $\lambda\mu$-calculus of call-by-value as a short-hand for the second order Church-style.

Our motivation comes from the observation that in Curry-style polymorphic calculi, control operators such as `callcc` or μ-operators cannot, in general, handle the terms placed on the control operator's left. In other words, control operators in Curry-style polymorphic calculi unexpectedly violate the eigenvariable condition of polymorphic generalization during the reductions, so that the subject reduction property no longer holds true in the system.

Following the continuation semantics, we also discuss the notion of values in classical system, and propose an extended form of values. It is shown that the CPS-translation is sound with respect to domain-free $\lambda 2$ (System F of Girard, Polymorphic calculus of Reynolds). We observe that the inverse of the soundness does not hold, and that adding $\perp$-reduction in Ong and Stewart [45] breaks down the soundness of the CPS-translation. By the non-trivial use of the modified CPS-translation, it can be obtained that the second order call-by-value $\lambda\mu$-calculus in domain-free style has the strong normalization property and the Church–Rosser property. Next, we study static properties of the call-by-value system; type checking, typability, and type inference. All the problems are proved undecidable for the second-order system in domain-free style [21], by a reduction from simple instances of the second-order unification problem [53, 54]. Finally, we give a brief comparison with standard ML plus `callcc`, and discuss a natural way to avoid the unsoundness of ML with `callcc` [26].

1. Styles of $\lambda 2$-terms; Curry-style, Church-style, and domain-free

There are two well-known styles of typed lambda calculi, *i.e.*, Curry-style and Church-style. Those styles are also called implicitly typed and explicitly typed, respectively. With respect to the simply typed lambda calculus $\lambda^{\to}$, there is a forgetful map from $\lambda^{\to}$ *à la* Church to *à la* Curry, and conversely, well-typed terms in $\lambda^{\to}$-Curry can be lifted to well-typed terms in $\lambda^{\to}$-Church [6]. In the case of ML [40], there also exists implicitly typed and explicitly typed systems, and they are essentially equivalent [29]. Hence, the implicitly typed system serves as a short-hand for the explicitly typed system.

However, the equivalence between Curry-style and Church-style does not always hold for complex systems. Parigot [46] introduced $\lambda\mu$-calculus in Curry-style as second order classical logic although $\lambda\mu$-calculus *à la* Church was also given [48]. An intrinsically classical reduction is called the structural reduction that is a kind

of permutative proof reductions in Prawitz [50] or the so-called commutative cut. The $\lambda\mu$-calculus of Parigot is now known as a call-by-name system. If we construct second-order $\lambda\mu$-calculus of call-by-value, then it will be clear that the Curry-style cannot work for a consistent system following the demonstration in this section.

In a call-by-value system of $\lambda\mu$, we can adopt one more permutative reduction [45,46], called the symmetric structural reduction[2], to handle the terms placed on the μ-operator's left as well. However, the application of the symmetric structural proof reduction in the Curry-style, in general, violates the eigenvariable condition of polymorphic generalization. Consider the following figure in which erasing type information from polymorphic terms dictates the uncorrect application of the symmetric structural reduction:

$$\dfrac{V : (\forall t.A_1) \to A_2 \quad \dfrac{\dfrac{M_1 : A_1}{[\alpha]M_1 : \bot; A_1^\alpha}}{\vdots}{M : \bot; A_1^\alpha}}{\dfrac{\mu\alpha.M : A_1}{\mu\alpha.M : \forall t.A_1}}}{V(\mu\alpha.M) : A_2} \quad \triangleright \quad \dfrac{\dfrac{V : (\forall t.A_1) \to A_2 \quad \dfrac{M_1 : A_1}{M_1 : \forall t.A_1}\,(\forall I)^*}{VM_1 : A_2}}{\dfrac{[\alpha](VM_1) : \bot; A_2^\alpha}{\vdots}{M[V \Rightarrow \alpha] : \bot; A_2^\alpha}}}{\mu\alpha.M[V \Rightarrow \alpha] : A_2}$$

where $M[V \Rightarrow \alpha]$ denotes a term obtained by replacing each subterm of the form $[\alpha]N$ in M with $[\alpha](VN)$. Here, assume that M is in the form of $[\alpha](\lambda y_1 \ldots y_n.M')$ and the type A_1 depends on type of some y_i $(1 \leq i \leq n)$. *I.e.*, $\lambda y_1 \ldots y_n.M'$ has type $A_1 \equiv B_1 \to \cdots \to B_n \to B$ where $y_i : B_i$, and t is a free type variable in B_i. Then the eigenvariable condition of $(\forall I)^*$ in the figure above is broken down, since the free type variable t in A_1 is still open under the assumption $y_i : B_i$. For instance,

$$\lambda x.(\lambda f.(\lambda x_1 x_2.x_2)(fx)(f(\lambda x.x)))(\mu\alpha.[\alpha](\lambda y.\mu\beta.[\alpha](\lambda v.y)))$$

has type $t \to t \to t$. But this term is reduced to $\lambda x.x$ by the use of the symmetric structural reduction. Let $P \equiv \lambda f.(\lambda x_1 x_2.x_2)(fx)(f(\lambda x.x))$ and $Q \equiv \mu\alpha.[\alpha](\lambda y.\mu\beta.[\alpha](\lambda v.y))$. Then the well-typed term

$$\lambda g.(\lambda x.g(PQx))(\lambda x.g(PQx)) : (\forall t'.(t' \to t')) \to t \to t$$

is similarly reduced to Curry's fixed point combinator, $\lambda g.(\lambda x.g(xx))(\lambda x.g(xx))$. On the other hand, the case $\mu\alpha.M$ of $\mu\alpha.[\alpha](\lambda v.M')$ and M' of $\mu\beta.[\alpha](\lambda x.x)$ is a special case where $v \notin FV(M') = FV(\mu\beta.[\alpha](\lambda x.x))$, and the symmetric

[2]Of course one can add the symmetric structural reduction to $\lambda\mu$ of call-by-name. However we could not expect the Church–Rosser property for such a system. Our explanation for the failure of the subject reduction is applicable to polymorphic calculi in Curry-style, including let-polymorphism, together with the symmetric structural reduction. See also [18].

structural reduction is applicable even to polymorphic $\mu\alpha.M$. For example,

$$\lambda x.((\lambda f.(\lambda x_1 x_2.x_2)(fx)(f(\lambda x.x)))(\mu\alpha.[\alpha](\lambda v.\mu\beta.[\alpha](\lambda x.x)))x) : t \to t$$

is type correctly reduced to $\lambda x.x$.

This kind of phenomenon with respect to `callcc` was first discovered by Harper and Lillibridge [26] as a counterexample for ML with `callcc`. The above examples show that the subject reduction property no longer holds true in the second-order Curry system with control operators. One can also find the similar phenomenon such that the second-order Curry system does not closed under η-reductions [39].

From the viewpoint of classical logic, the fatal defect in type preservation under reductions can be explained as follows: in $\lambda\mu$-calculus *à la* Curry (second-order classical logic), an application of the symmetric structural reduction, in general, breaks down the eigenvariable condition of polymorphic generalization, and hence erasing polymorphic type information makes typable terms unclosed under the symmetric structural reductions. As a result, the contexts placed on the polymorphic μ-operator's left cannot be handled correctly by the symmetric structural reduction, *i.e.*, the failure of the subject reduction property. In terms of explicit polymorphism, in other words, an evaluation under Λ-abstractions cannot be allowed without restricting $\Lambda t.M$ to $\Lambda t.V$ [27] where V is called a value. In the example above, however, the polymorphic term Q cannot be considered as a value. Even in the Damas–Milner style [11] (implicitly typed ML) plus control operators, a similar defect still happens under an ML-like call-by-value [27, 28].

To avoid such a problem in implicitly typed ML with control operators, one can adopt an η-like expansion for polymorphic control operators [18], such that

$$\texttt{let}\, f = \mu\alpha.M_1 \;\texttt{in}\; M_2 \;\triangleright\; \texttt{let}\, f = \lambda x.\mu\alpha.M_1[\alpha \Leftarrow x] \;\texttt{in}\; M_2,$$

where each subterm in the form of $[\alpha](\lambda y.w)$ in M_1 is replaced with $[\alpha](\lambda y.w)x$.

Another natural way to avoid the problem is to take a domain-free system introduced formally by Barthe and Sørensen [7], Table 1.

TABLE 1. Styles of (typed) $\lambda 2$-terms.

	object var. abst.	type var. abst.	type app.
Church-style	$\lambda x:A.M$	$\Lambda t.M$	MA
Domain free	$\lambda x.M$	$\Lambda t.M$	MA
Curry-style	$\lambda x.M$		

In the above example, the term Q is a polymorphic term, and this type becomes $\forall t.(t \to t)$. Here, the explicitly typed term as a form of a value, $V \equiv \Lambda t.Q$ is used

for β_v-reductions, such that

$$\lambda x.(\lambda f.(\lambda x_1 x_2.x_2)(ftx)(f(t \to t)(\lambda x.x)))\ V : t \to t \to t$$

is now reduced to $\lambda v.\lambda x.x$.

In the next section, under the call-by-value strategy we introduce a domain-free $\lambda\mu$-calculus, which is regarded as a short-hand for the complete Church-style. To obtain the results in this paper, it is enough to consider a system such that $\Lambda t.M$ is represented simply as ΛM such as a lifting and MA as $M()$, and $(\Lambda M)()$ is reduced to M. A similar observation is given for let-polymorphism by name in Leroy [36]. The annotations Λ and $()$ for polymorphic terms play a role of choosing an appropriate computation under call-by-value. However, from the viewpoint of classical logic, a domain-free $\lambda\mu$-calculus is considered here rather than such a simplified polymorphism using the annotations.

On the other hand, Harper and Lillibridge [27] extensively studied explicit polymorphism and CPS-conversion for F_w with `callcc`. The call-by-value system $\lambda_V\mu$ introduced in Section 2 can be regarded as a meaningful simplification of the second order fragment of their system.

2. $\lambda_V\mu$-CALCULUS IN DOMAIN-FREE STYLE

Following the observation in the previous section, we introduce $\lambda_V\mu$-calculus: a domain-free $\lambda\mu$-calculus of call-by-value[3] for polymorphism, *i.e.*, second order λ_V plus μ-operator in domain-free style. Terms in domain-free style have domain-free λ-abstraction [7] and domain-free μ-abstraction.

The types A are defined from type variables t and a type constant $\bot$. The negation $\neg A$ is defined as $A \to \bot$.

Definition 2.1 (*Types A*). $A ::= t \mid \bot \mid A \to A \mid \forall t.A$.

We have a set of term variables $x, y, z, \ldots$, and a set of names (that will be called continuation variables later) $\alpha, \beta, \ldots$. The type assumptions are defined as usual, and Δ is used for a set of name-indexed types.

Definition 2.2 (*Type Assumptions* Γ, Δ). $\Gamma ::= \langle\,\rangle \mid x{:}A, \Gamma$; $\Delta ::= \langle\,\rangle \mid A^\alpha, \Delta$.

The terms M are defined as term variables, λ-abstractions, applications, μ-abstractions, or named terms. Since we have sorted variables, *i.e.*, term variable x and type variable t, we have explicit distinction between terms and types, and then λ-abstraction is used for both term variable and type variable abstractions.

Definition 2.3 (*Terms M*). $M ::= x \mid \lambda x.M \mid MM \mid \lambda t.M \mid MA \mid \mu\alpha.M \mid [\alpha]M$.

From a logical viewpoint, the typing rule $(\bot E)$ for $\mu\alpha.M$ is regarded as a classical inference rule such that infer $\Gamma, \neg\Delta \vdash \mu\alpha.M : A$ from $\Gamma, \neg\Delta, \alpha : \neg A \vdash$

[3]The system $\lambda_V\mu$ is also an extended system of call-by-value $\lambda\mu$ in [16].

$M : \bot$. The typing rule $(\bot I)$ for $[\alpha]M$ can be considered as a special case of $\bot$-introduction by the use of $(\to E)$. On the basis of the continuation semantics in the next section, a name can be interpreted as a continuation variable. In the rule $(\bot I)$, the continuation variable α appears only in the function-position, but not in the argument-position. Here, the negative assumption $\alpha : \neg A$ corresponding to A^α of $(\bot I)$ can be discharged only by $(\bot E)$. This style of proofs consisting of the special case of $\bot$-introduction is called a regular proof in Andou [1].

Definition 2.4 (*Type Assignment Rules*).

$$\Gamma \vdash x : \Gamma(x); \Delta$$

$$\frac{\Gamma \vdash M_1 : A_1 \to A_2; \Delta \Gamma \vdash M_2 : A_1; \Delta}{\Gamma \vdash M_1 M_2 : A_2; \Delta} (\to E) \qquad \frac{\Gamma, x{:}A_1 \vdash M : A_2; \Delta}{\Gamma \vdash \lambda x.M : A_1 \to A_2; \Delta} (\to I)$$

$$\frac{\Gamma \vdash M : \forall t.A_1; \Delta}{\Gamma \vdash M A_2 : A_1\,[t := A_2]\,; \Delta} (\forall E) \qquad \frac{\Gamma \vdash M : A; \Delta}{\Gamma \vdash \lambda t.M : \forall t.A; \Delta} (\forall I)^*$$

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

where $(\forall I)^*$ denotes the eigenvariable condition.

The notion of values is introduced below as an extended form; the class of values is closed under both value-substitutions induced by (β_v) and left and right context-replacements induced by $(\mu_{l,r})$, as defined later.

Definition 2.5 (*Values V*). $V ::= x \mid \lambda x.M \mid \lambda t.M \mid [\alpha]M$.

The definition of the reduction rules is given below under call-by-value. In particular, the classical reductions $(\mu_{l,r,t})$ below can be explained as a logical permutative reduction in the sense of Prawitz [50, 51] and Andou [1–3]. Here, in the reduction of $(\mu\alpha.M)N \triangleright \mu\alpha.M[\alpha \Leftarrow N]$, since both type of $\mu\alpha.M$ and type of each subterm M' with the form $[\alpha]M'$ in M can be considered as members of the segments ending with the type of $\mu\alpha.M$, the application of $(\to E, \forall E)$ is shifted up to each occurrence M', and then $M[y \Leftarrow N]$ (each $[\alpha]M'$ is replaced with $[\alpha](M'N)$) is obtained. This reduction is also called a structural reduction in Parigot [46]. On the other hand, since a term of the form $\mu\alpha.M$ is not regarded as a value, $(\lambda x.M_1)(\mu\alpha.M_2)$ will not be a β-contractum, but will be a contractum of (μ_l) below, which can be considered as a symmetric structural reduction. $FV(M)$ stands for the set of free variables in M, and $FN(M)$ for the set of free names in M.

Definition 2.6 (*Term Reductions*).

(β_v) $(\lambda x.M)V \rhd M[x := V]$

(η_v) $\lambda x.Vx \rhd V$ if $x \notin FV(V)$

(β_t) $(\lambda t.M)A \rhd M[t := A]$

(η_t) $\lambda t.Vt \rhd V$ if $t \notin FV(V)$

(μ_t) $(\mu\alpha.M)A \rhd \mu\alpha.M[\alpha \Leftarrow A]$

(μ_l) $(\mu\alpha.M_1)M_2 \rhd \mu\alpha.M_1[\alpha \Leftarrow M_2]$

(μ_r) $V(\mu\alpha.M) \rhd \mu\alpha.M[V \Rightarrow \alpha]$

(rn) $[\alpha](\mu\beta.V) \rhd V[\beta := \alpha]$

$(\mu\text{-}\eta)$ $\mu\alpha.[\alpha]M \rhd M$ if $\alpha \notin FN(M)$

where the term $M[\alpha \Leftarrow N]$ denotes a term obtained by M replacing each subterm of the form $[\alpha]M'$ in M with $[\alpha](M'N)$. That is, the terms (context) placed on $\mu\alpha.M$'s right is replaced in an argument position of M' in $[\alpha]M'$. In turn, the term $M[V \Rightarrow \alpha]$ denotes a term obtained by M replacing each subterm of the form $[\alpha]M'$ in M with $[\alpha](VM')$.

Our notion of values is closed under the reductions, *i.e.*, values are reduced to simpler values. Because eta-reductions and renaming rule (rn) are restricted to the extended values. The restriction of the two rules to values is essentially necessary to establish a sound CPS-translation in Section 3. We note that as observed in Ong and Stewart [45], there are closed normal forms which are not values, called canonical forms, *e.g.*, $\mu\alpha.[\alpha](\lambda x.\mu\beta.[\alpha](\lambda v.x))$. Those terms can be reduced by (S_3) in [47] or $\zeta^{\mathsf{ext}}_{\mathsf{fun}}$ in [45], but in this case, $(\mu\alpha.M)(\mu\beta.N)$ is reduced in the two ways (not confluent). Note also that the failure of operational extensionality for $\mu\mathrm{PCF}^-_v$ is demonstrated in [45]. In fact, $\zeta^{\mathsf{ext}}_{\mathsf{fun}}$ becomes admissible under the eta-reduction and (μ_r). In this paper, however a term in the form of $\mu\alpha.M$ is not a value, and we have the value-restricted (η_v) instead of the eta-reduction itself.

We denote $\rhd_\mu$ by the one-step reduction induced by $\rhd$. We write $=_\mu$ for the reflexive, symmetric, and transitive closure of $\rhd_\mu$. The notations such as $\rhd_\beta$, $\rhd_{\beta\eta}$, $\rhd^+_\beta$, $\rhd^*_{\beta\eta}$, $=_{\beta\eta}$, etc. are defined as usual ($+$ for the transitive closure, and $*$ for the reflexive and transitive closure), and $\rhd^i_\beta$ denotes i-step β-reductions ($i \geq 0$).

On the basis of the sorted variables; term variable x, name α, and type variable t, we have explicit distinction between terms and types; term-applications and type-applications; and term variable-abstractions and type variable-abstractions. Hence, when a well-typed term is given, the corresponding type assignment rule

is uniquely determined by the shape of the term. From this syntactical property of terms, we have the natural generation lemma for $\lambda_V\mu$.

Lemma 2.7 (*Generation Lemma for* $\lambda_V\mu$).

(1) *If* $\Gamma \vdash x : A; \Delta$, *then* $\Gamma(x) = A$.
(2) *If* $\Gamma \vdash M_1 M_2 : A; \Delta$, *then* $\Gamma \vdash M_1 : A_1 \to A; \Delta$ *and* $\Gamma \vdash M_2 : A_1; \Delta$ *for some* A_1.
(3) *If* $\Gamma \vdash \lambda x.M : A; \Delta$, *then* $\Gamma, x{:}A_1 \vdash M : A_2; \Delta$ *and* $A \equiv A_1 \to A_2$ *for some* A_1 *and* A_2.
(4) *If* $\Gamma \vdash MA_1 : A; \Delta$, *then* $\Gamma \vdash M : \forall t.A_2; \Delta$ *and* $A \equiv A_2[t := A_1]$ *for some* A_2.
(5) *If* $\Gamma \vdash \lambda t.M : A; \Delta$, *then* $\Gamma \vdash M : A_1; \Delta$ *and* $A \equiv \forall t.A_1$ *and* $t \notin FV(\Gamma)$ *for some* A_1.
(6) *If* $\Gamma \vdash [\alpha]M : A; \Delta$, *then* $\Gamma \vdash M : A_1; \Delta_1$ *and* $A \equiv \bot$ *and* $\Delta \equiv \Delta_1, A_1^\alpha$ *for some* A_1.
(7) *If* $\Gamma \vdash \mu\alpha.M : A; \Delta$, *then* $\Gamma \vdash M : \bot; \Delta, A^\alpha$.

Proposition 2.8 (*Subject Reduction Property for* $\lambda_V\mu$). *If we have* $\Gamma \vdash M_1 : A; \Delta$ *and* $M_1 \rhd_\mu M_2$ *in* $\lambda_V\mu$, *then* $\Gamma \vdash M_2 : A; \Delta$ *in* $\lambda_V\mu$.

Proof. By induction on the derivation of $M_1 \rhd_\mu M_2$. Note that in $\lambda_V\mu$, typing rules are uniquely determined depending on the shape of terms. □

The well-known type erasure $|M|$ is defined as follows:

$$|x| = x \qquad |\lambda x.M| = \lambda x.|M| \qquad |M_1 M_2| = |M_1|\,|M_2|$$
$$|\lambda t.M| = |M| \qquad |MA| = |M| \qquad |\mu\alpha.M| = \mu\alpha.|M| \qquad |[\alpha]M| = [\alpha]|M|.$$

Then it can be seen that the typing relation is preserved between $\lambda_V\mu$ and implicitly typed $\lambda\mu$:

(i) if we have $\Gamma \vdash M : A; \Delta$ in $\lambda_V\mu$, then $\Gamma \vdash |M| : A; \Delta$ in implicit $\lambda\mu$;
(ii) if we have $\Gamma \vdash M_1 : A; \Delta$ in implicit $\lambda\mu$, then there exists M_2 such that $M_1 = |M_2|$ and $\Gamma \vdash M_2 : A; \Delta$ in $\lambda_V\mu$.

The set of types inhabited by terms coincides between implicit $\lambda\mu$ and $\lambda_V\mu$. However, erasing type information makes typable terms unclosed under reductions, such as η-reduction of the erasure in Mitchell [39], and hence the subject reduction property for $|M|$ is broken down. See counterexamples in Section 1.

3. CPS-TRANSLATIONS FOR $\lambda_V\mu$-CALCULUS

3.1. SOUNDNESS OF THE CPS-TRANSLATION

To provide the CPS-translation, we define a domain-free $\lambda 2$ (see also [7]) as the intuitionistic fragment of $\lambda_V\mu$. Here, we have two kinds of term variables; besides λ-variables $x, y, z, \ldots$ used in λ-calculus as usual, the system $\lambda 2$ has the distinguished variables $\alpha, \beta, \ldots$ called continuation variables. Reduction rules in

domain-free $\lambda 2$ are also defined as usual under call-by-name. The term with the form $[\alpha]M$ (value) will be interpreted as $\lambda k.k(\overline{M}\alpha)$, where the representation of $\overline{M}\alpha$ is consumed by the continuation k, such as the case of λ-abstraction. The translation from $\lambda_V\mu$ to domain-free $\lambda 2$, with an auxiliary function Ψ for values, comes from Plotkin [49].

Definition 3.1 (*CPS-Translation*).

(i) $\overline{x} = \lambda k.kx$.
(ii) $\overline{\lambda x.M} = \lambda k.k(\lambda x.\overline{M})$.
(iii) $\overline{\lambda t.M} = \lambda k.k(\lambda t.\overline{M})$.
(iv) $\overline{[\alpha]M} = \lambda k.k(\overline{M}\alpha)$.
(v) $\overline{\mu\alpha.M} = \lambda\alpha.\overline{M}(\lambda x.x)$.
(vi) $\overline{MA} = \lambda k.\overline{M}(\lambda m.mA^qk)$.
(vii) $\overline{M_1M_2} = \lambda k.\overline{M_1}(\lambda m.\overline{M_2}(\lambda n.mnk))$.

(a) $\Psi(x) = x$.
(b) $\Psi(\lambda x.M) = \lambda x.\overline{M}$.
(c) $\Psi(\lambda t.M) = \lambda t.\overline{M}$.
(d) $\Psi([\alpha]M) = \overline{M}\alpha$.

① $t^q = t$.
② $(A_1 \to A_2)^q = A_1^q \to \neg\neg A_2^q$.
③ $(\forall t.A)^q = \forall t.\neg\neg A^q$.

According to the continuation semantics of Meyer and Wand [38], our definition of the CPS-translation can be read as follows: if we have a variable x, then the value x is passed on to the continuation k. In the case of a λ-abstraction, a certain function that will take two arguments is passed on to the continuation k. If we have a term with a continuation variable α, then a certain function with the argument α is passed on to the continuation k, where the variable α will be substituted by a continuation. Here, it would be natural that a value is regarded as the term that is mapped by Ψ to some term consumed by the continuation k, since the continuation is the context in which a term is evaluated and then to which the value is sent. Our notion of values as an extended form is derived following this observation.

Lemma 3.2. *Let* $=$ *denote the definitional equality of the CPS-translation.*

(1) *For any term* M *where* $k \notin FV(M)$, $\lambda k.\overline{M}k \rhd_\beta \overline{M}$.
(2) *For any value* V, $\overline{V} = \lambda k.k\Psi(V)$.
(3) *For any term* M, *value* V, *and type* A,
we have $\overline{M[x := V]} = \overline{M}[x := \Psi(V)]$ *and* $\overline{M[t := A]} = \overline{M}[t := A^q]$.

The above lemma can be proved by straightforward induction. On the basis of the CPS-translation, the left and right context-replacements $M[\alpha \Leftarrow M_1]$ and $M[V \Rightarrow \alpha]$ can be interpreted as the following substitutions for continuation variables, respectively.

Lemma 3.3. *Let M contain i free occurrences of $[\alpha]$ where $i \geq 0$.*

Then we have that $\overline{M[\alpha \Leftarrow M_1]} \rhd_\beta^i \overline{M}[\alpha := \lambda m.\overline{M_1}(\lambda n.mn\alpha)]$ and $\overline{M[\alpha \Leftarrow A]} \rhd_\beta^i \overline{M}[\alpha := \lambda m.mA^q\alpha]$.

Proof. By induction on the structure of M. See Appendix 5.1. □

Lemma 3.4. *For any term M and value V, $\overline{M[V \Rightarrow \alpha]} \rhd_\beta^{3i} \overline{M}[\alpha := \lambda n.\Psi(V)n\alpha]$, where M contains i free occurrences of $[\alpha]$.*

Proof. By induction on the structure of M. See Appendix 5.2. □

Let $\rhd_{\beta\eta r}$ be one-step $\rhd_\mu$ consisting of (β_v), (β_t), (η_v), (η_t), $(\mu\text{-}\eta)$, or (rn). Let $\rhd_{st}$ be one-step $\rhd_\mu$ consisting of (μ_l), (μ_r), or (μ_t).

Lemma 3.5. *If we have $M \rhd_\mu N$ in $\lambda_V\mu$, then $\overline{M} =_{\beta\eta} \overline{N}$ in domain-free $\lambda 2$. To be precise, the following holds true:*

(1) *if $M \rhd_{\beta\eta r} N$ in $\lambda_V\mu$, then $\overline{M} \rhd_{\beta\eta}^+ \overline{N}$ in domain-free $\lambda 2$;*

(2) *if $M \rhd_{st} N$ in $\lambda_V\mu$, then $\overline{M} =_\beta \overline{N}$ in domain-free $\lambda 2$.*

Proof. By induction on the derivation of $M \rhd_\mu N$. See Appendix 5.3. □

Now, we have confirmed the soundness of the translation in the sense that equivalent $\lambda_V\mu$-terms are translated into equivalent domain-free $\lambda 2$-terms. This property essentially holds for untyped terms.

Proposition 3.6 (*Soundness of the CPS-Translation*). *If we have $M =_\mu N$ in $\lambda_V\mu$, then then $\overline{M} =_{\beta\eta} \overline{N}$ in domain-free $\lambda 2$.*

The translation logically establishes the double negation translation of Kuroda. For a set of name-indexed formulae Δ, we define $\neg(A^\alpha, \Delta)^q$ as $\alpha : \neg A^q, \neg\Delta^q$.

Proposition 3.7 (*Kuroda Translation*). *If $\lambda_V\mu$ has $\Gamma \vdash M : A; \Delta$, then domain-free $\lambda 2$ has $\Gamma^q, \neg\Delta^q \vdash \overline{M} : \neg\neg A^q$.*

Proof. By induction on the derivation. □

From the consistency of domain-free $\lambda 2$, it is derived that $\lambda_V\mu$ is consistent in the sense that there is no closed term M such that $\vdash M : \bot$; in $\lambda_V\mu$.

With respect to Proposition 3.6, it is known that the implication is, in general, not reversible. The counterexample in [49] is not well-typed. Even though we consider well-typed $\lambda_V\mu$-terms, the completeness does not hold for $\lambda_V\mu$: if we have $M_1 \equiv (\lambda x.x)(xy)$ and $M_2 \equiv xy$ in $\lambda_V\mu$, then $\overline{M_1} =_{\beta\eta} xy =_{\beta\eta} \overline{M_2}$ in $\lambda 2$, but $M_1 \neq_\mu M_2$ in $\lambda_V\mu$. Note that in this counterexample, if one excluded η-reduction, then $\overline{M_1} \neq_\beta \overline{M_2}$. Following Hofmann [30], the rewrite rules of $\lambda_V\mu$ are weak from the viewpoint of the semantics, since `Ident`, $(\lambda x.x)M = M$ is necessary in this case.

According to Ong and Stewart [45], their call-by-value $\lambda\mu$-calculus has more reduction rules with the help of type annotation; $\bot$-reduction:

$$V^{\bot\to A}M^\bot \rhd \mu\beta^A.M^\bot \quad \text{if} \quad A \not\equiv \bot.$$

Here, assume that we have $N_1 \equiv (\lambda x.x)(x([\alpha]y))$ and $N_2 \equiv x([\alpha]y)$, such that $x : \bot \to A, y : A \vdash N_i : A; A^{\alpha}$ $(i = 1, 2)$ where $A \not\equiv \bot$ in $\lambda_V\mu$. Then N_1 and N_2 are reduced to $N_3 \equiv \mu\beta.[\alpha]y$ by the use of $\bot$-reduction. Now, we have $\overline{N_1} =_{\beta\eta} x(\alpha y) =_{\beta\eta} \overline{N_2}$ in $\lambda 2$, but $\overline{N_3} =_{\beta} \lambda\beta.\alpha y$ in $\lambda 2$. This example means that the soundness of the CPS-translation is broken down for $\lambda_V\mu$ with $\bot$-reduction, even in the absence of η-reduction. However, on the basis of the correspondence between μ-operator and Felleisen's $\mathcal{C}$-operator [14] such that $\mu\alpha.M = \mathcal{C}(\lambda\alpha.M)$ and $[\alpha]M = \alpha M$, one obtains that $x(\alpha y) =_{\mathcal{C}} (\lambda x.\mathcal{A}(x))(\alpha y) =_{\mathcal{C}} \mathcal{A}(\alpha y) =_{\mathcal{C}} \mathcal{C}(\lambda\beta.\alpha y)$ in the equational theory $\lambda_{\mathcal{C}}$ [30]. From the naive observation, Hofmann's categorical models for $\lambda_{\mathcal{C}}$ would also work for an equational version of the call-by-value $\lambda\mu$-calculus.

3.2. Strong normalization and Church–Rosser for $\lambda_V\mu$-calculus

In order to demonstrate the strong normalization for $\lambda_V\mu$, we give a modification of the previous CPS-translation, called a modified CPS-translation such as in [24, 48, 49].

Definition 3.8 (*Modified CPS-Translation*).

(i) $\overline{\overline{x}} = kx$.

(ii) $\overline{\overline{\lambda x.M}} = k(\lambda x.\lambda k.\overline{\overline{M}})$.

(iii) 1) $\overline{\overline{V_1V_2}} = \Phi(V_1)\Phi(V_2)k$.

2) $\overline{\overline{VM}} = \overline{\overline{M}}[k := \lambda n.\Phi(V)nk]$.

3) $\overline{\overline{MV}} = \overline{\overline{M}}[k := \lambda m.m\Phi(V)k]$.

4) $\overline{\overline{M_1M_2}} = \overline{\overline{M_1}}[k := \lambda m.\overline{\overline{M_2}}[k := \lambda n.mnk]]$.

(iv) $\overline{\overline{\lambda t.M}} = k(\lambda t.\lambda k.\overline{\overline{M}})$.

(v) 1) $\overline{\overline{VA}} = \Phi(V)A^qk$.

2) $\overline{\overline{MA}} = \overline{\overline{M}}[k := \lambda m.mA^qk]$.

(vi) $\overline{\overline{[\alpha]M}} = k(\overline{\overline{M}}[k := \alpha])$.

(vii) $\overline{\overline{\mu\alpha.M}} = \overline{\overline{M}}[k := \lambda x.x][\alpha := k]$.

(a) $\Phi(x) = x$.

(b) $\Phi(\lambda x.M) = \lambda x.\lambda k.\overline{\overline{M}}$.

(c) $\Phi(\lambda t.M) = \lambda t.\lambda k.\overline{\overline{M}}$.

(d) $\Phi([\alpha]M) = \overline{\overline{M}}[k := \alpha]$.

The modified CPS-translation has the following properties:

(1) $\overline{\overline{V}} = k\Phi(V)$.

(2) $\overline{\overline{M}}[x := \Phi(V)] = \overline{\overline{M[x := V]}}$.

(3) $\overline{\overline{M}}[t := A^q] = \overline{\overline{M[t := A]}}$.

Lemma 3.9. *We have the following property with respect to replacements:*

(1) $\overline{\overline{M[\alpha \Leftarrow N]}} = \overline{\overline{M}}[\alpha := \lambda m.\overline{\overline{N}}[k := \lambda n.mn\alpha]]$.

(2) $\overline{\overline{M[\alpha \Leftarrow A]}} = \overline{\overline{M}}[\alpha := \lambda m.mA^q\alpha]$.
(3) $\overline{\overline{M[V \Rightarrow \alpha]}} = \overline{\overline{M}}[\alpha := \lambda n.\Phi(V)n\alpha]$.

Proof. By induction on the structure of M. See Appendix 5.4. □

Now the modified CPS-translation can also establish the soundness from the following property.

Lemma 3.10. *If we have $M_1 \rhd_\mu M_2$ in $\lambda_V\mu$, then $\overline{\overline{M_1}} \rhd^*_{\beta\eta} \overline{\overline{M_2}}$ in domain-free $\lambda 2$. To be precise, the following holds true:*

(1) *if $M_1 \rhd_{\beta\eta r} M_2$ in $\lambda_V\mu$, then $\overline{\overline{M_1}} \rhd^*_{\beta\eta} \overline{\overline{M_2}}$ in domain-free $\lambda 2$.*
More precisely, let M_1 contain no vacuous μ-abstraction, i.e., $\alpha \in FN(M)$ for every subterm $\mu\alpha.M$ of M_1. If $M_1 \rhd_{\beta\eta r} M_2$, then $\overline{\overline{M_1}} \rhd^+_{\beta\eta} \overline{\overline{M_2}}$ in domain-free $\lambda 2$;
(2) *if $M_1 \rhd_{st} M_2$ in $\lambda_V\mu$, then $\overline{\overline{M_1}} = \overline{\overline{M_2}}$ in domain-free $\lambda 2$.*

Proof. By induction on the derivation of $M_1 \rhd_\mu M_2$. See Appendix 5.5 and also the following remarks. □

It is remarked that a crucial case[4] of (1) above is illustrated as follows: assume that $\alpha \notin FN(M)$. Even though we have $M_1 \rhd_{\beta\eta r} M_2$ and then $\overline{\overline{M_1}} \rhd^+_{\beta\eta} \overline{\overline{M_2}}$, one cannot have $\overline{\overline{(\mu\alpha.M)M_1}} \rhd^+_{\beta\eta} \overline{\overline{(\mu\alpha.M)M_2}}$. In fact, this illustration gives that $\overline{\overline{(\mu\alpha.M)M_1}} = \overline{\overline{\mu\alpha.M}} = \overline{\overline{(\mu\alpha.M)M_2}}$ since $\alpha \notin FN(M)$. On the other hand, Lemma 3.5 says that $\overline{(\mu\alpha.M)M_1} \rhd^+_{\beta\eta} \overline{(\mu\alpha.M)M_2}$ still holds even under the same assumption.

Assume that a set of reduction rules is divided into two groups; one contains reductions rules which can be interpreted as a transitive closure under a translation, *e.g.* $\overline{\overline{(\lambda x.M)V}} \rhd^+ \overline{\overline{M[x := V]}}$ for $(\lambda x.M)V \rhd M[x := V]$, and another has rules which can be interpreted as a reflexive closure, *e.g.* $\overline{\overline{(\mu\alpha.M)N}} = \overline{\overline{\mu\alpha.M[\alpha \Leftarrow N]}}$ for $(\mu\alpha.M)N \rhd \mu\alpha.M[\alpha \Leftarrow N]$, and $\overline{\overline{(\mathtt{raise}(M))N}} = \overline{\overline{\mathtt{raise}(M)}}$ for $(\mathtt{raise}(M))N \rhd \mathtt{raise}(M)$. If a subterm in the left-hand side of a reduction rule in the latter group can disappear in the right-hand side, *e.g.*, the subterm N in the left-hand side has neither occurrence in the right-hand side $\mathtt{raise}(M)$ nor in $\mu\alpha.M[\alpha \Leftarrow N]$ if $\alpha \notin FN(M)$, then the crucial case happens in general. That is, one step rewriting by a rule in the former group becomes not transitive but reflexive and transitive under the translation.

The modified CPS-translation also gives an intuitionistic proof of formulae embedded by the double negation translation of Kuroda in the following sense.

Proposition 3.11 (*Kuroda Translation*). *If $\lambda_V\mu$ has $\Gamma \vdash M : A; \Delta$, then domain-free $\lambda 2$ has $\Gamma^q, \neg\Delta^q, k{:}\neg A^q \vdash \overline{\overline{M}} : \bot$.*

Proof. By induction on the derivation. □

[4]Even the literature [24, 48] seems unaware of this crucial case.

As observed from Propositions 3.7 and 3.11, one can find an intimate relation between the two CPS-translations:

Proposition 3.12. *Let M be a $\lambda_V\mu$-term and V be a value. Then for any term K, we have the following property:*
(i) $\Psi(V) \rhd_\beta^* \Phi(V)$, (ii) $\overline{M}K \rhd_\beta^+ \overline{\overline{M}}[k := K]$, *and* (iii) $\overline{M} \rhd_\beta^* \lambda k.\overline{\overline{M}}$.

Proof. By simultaneous induction on the structures of V and M. □

To demonstrate the strong normalization for any well-typed $\lambda_V\mu$-term, we first show that well-typed $\lambda_V\mu$-terms without vacuous μ-abstractions are strongly normalizable. From Lemma 3.10, Proposition 3.11, and the fact that domain-free $\lambda 2$ is strongly normalizing [7], the possibility of the infinite reduction path may happen only in the case (2) of Lemma 3.10. This means that we should prove the strong normalization for $\rhd_{st}$.

Proposition 3.13 (*Strong Normalization of $\rhd_{st}$*). *Any $\lambda_V\mu$-term is strongly normalizable with respect to the structural reductions $\rhd_{st}$.*

Proof. From Lemma 5.7 in Appendix 5.6. □

At the moment we have almost, but not completely, obtained our main result.

Lemma 3.14. *Any well-typed $\lambda_V\mu$-term that has no vacuous μ-abstractions is strongly normalizable.*

Proof. Suppose that a well-typed $\lambda_V\mu$-term M is not strongly normalizable with respect to $\rhd_\mu$. Then there exists an infinite reduction path σ from M; $M \rhd_\mu M_1 \rhd_\mu M_2 \rhd_\mu \cdots$. Here, we have three cases for the infinite reduction path σ.

Case 1. The infinite sequence σ consists only of $\rhd_{\beta\eta r}$: we now have $M \rhd_{\beta\eta r} M_1 \rhd_{\beta\eta r} M_2 \rhd_{\beta\eta r} \cdots$. From Lemma 3.5 (1), we also have an infinite reduction path $\overline{M} \rhd_{\beta\eta}^+ \overline{M_1} \rhd_{\beta\eta}^+ \overline{M_2} \rhd_{\beta\eta}^+ \cdots$. On the other hand, $\overline{M}$ is well-typed from Proposition 3.7, and hence $\overline{M}$ is strongly normalizable. Now we have a contradiction.

Case 2. The infinite sequence σ consists only of $\rhd_{st}$: in this case, we have $M \rhd_{st} M_1 \rhd_{st} M_2 \rhd_{st} \cdots$, which contradicts to Lemma 3.13 (strong normalizability with respect to $\rhd_{st}$).

Case 3. The infinite sequence σ consists of alternate $\rhd_{\beta\eta r}^+$ and $\rhd_{st}^+$: assume that we have $M \rhd_{st}^+ M_1 \rhd_{\beta\eta r}^+ M_2 \rhd_{st}^+ M_3 \rhd_{\beta\eta r}^+ \cdots$. Since we have $\alpha \in FN(M')$ for each subterm $\mu\alpha.M'$ of M, Lemma 3.10 proves that $\overline{\overline{M}} = \overline{\overline{M_1}} \rhd_{\beta\eta}^+ \overline{\overline{M_2}} = \overline{\overline{M_3}} \rhd_{\beta\eta}^+ \cdots$, which gives an infinite reduction path from $\overline{\overline{M}} = \overline{\overline{M_1}}$. On the other hand, $\overline{\overline{M}} = \overline{\overline{M_1}}$ is well-typed from Proposition 3.11 and hence strongly normalizable, which is a contradiction. □

A context with a hole $[\,]$, denoted by $\mathcal{E}$ or $\mathcal{F}$, is defined as usual:

Definition 3.15 (*Context* $\mathcal{E}$). $\mathcal{E} ::= [\,] \mid \mathcal{E}M \mid V\mathcal{E} \mid \mathcal{E}A$.

A term $\mathcal{E}[\mu\alpha.M]$ is also represented as $\mathcal{E}_n[\mathcal{E}_{n-1}[\cdots\mathcal{E}_1[\mu\alpha.M]\cdots]]$ where $\mathcal{E}_i[\,]$ is either $[\,]M_i$, $V_i[\,]$, or $[\,]A_i$, and for simplicity, $\mathcal{E}_i$ also denotes the terms M_i, V_i, or the type A_i.

There remains to be proved later Case 3 of Lemma 3.14 where M contains vacuous μ-abstractions[5] and $\overline{\overline{M}}$ admits no infinite reduction path. From Lemma 3.10, one has that if $\overline{\overline{M_1}} \not\rhd^+_{\beta\eta} \overline{\overline{M_2}}$ then either $M_1 \not\rhd_{\beta\eta r} M_2$ or there exists a subterm $\mu\alpha.M'$ of M such that $\alpha \notin FN(M')$. It is noted that if $M_1 \rhd_{\beta\eta r} M_2$ in σ such that $\overline{\overline{M_1}} \not\rhd^+_{\beta\eta} \overline{\overline{M_2}}$, then M_1 contains a subterm of the form $\mu\alpha.M'$ where $\alpha \notin FN(M')$. Moreover, $M_1 \rhd_{\beta\eta r} M_2$ and $\overline{\overline{M_1}} \not\rhd^+_{\beta\eta} \overline{\overline{M_2}}$ hold only when M_1 contains a strict subterm N such that $M_1 \rhd^+_{st} M$, $M_2 \rhd^+_{st} M$, and M has no occurrence of N for some M, and that the reduction $M_1 \rhd_{\beta\eta r} M_2$ is executed by reducing the subterm N, see Lemma 3.19 below. In fact, even when one does not apply structural reductions such as $\mathcal{E}[\mu\alpha.M'] \rhd^+_{st} \mu\alpha.M'$ in σ, we have $\overline{\overline{\mathcal{E}[\mu\alpha.M']}} = \overline{\overline{\mu\alpha.M'}}$. Since $\overline{\overline{M}}$ is now terminating although M admits an infinite reduction path, there exists such a context $\mathcal{E}$ in the terms of the finite reduction path from $\overline{\overline{M}}$. Moreover, the infinite reduction path σ essentially comes from a subterm of M, which is reduced to $\mathcal{E}_i$ of $\mathcal{E}$ whose image disappears under the modified CPS-translation together with vacuous μ-abstraction.

Given a $\lambda_V\mu$-term M, we define a finite set of subterms of M such that the subterm may have an infinite reduction sequence but the image of the subterm vanishes under the modified CPS-translation.

Definition 3.16 (*Set of Vanishing Subterms*; $\mathcal{V}term$, $\mathcal{VT}$).

$$\mathcal{V}term(M) = \left\{ \mathcal{E}_i \;\middle|\; \begin{array}{l} M \rhd^*_{st} M_0 \text{ and } M_0 \text{ contains a subterm } \mathcal{E}[\mu\alpha.M'] \\ \text{where } \alpha \notin FN(M') \text{ and } \mathcal{E}[\,] \equiv \mathcal{E}_n[\mathcal{E}_{n-1}[\cdots\mathcal{E}_1[\,]\cdots]] \\ \text{with term } \mathcal{E}_i \text{ for } 1 \le i \le n \end{array} \right\}.$$

A set of vanishing subterms of M, denoted by $\mathcal{VT}(M)$, is defined as follows:

$\mathcal{VT}(M) = \{N \mid N \text{ is a subterm of } N' \in \mathcal{V}term(M)\}$.

Lemma 3.17. *For any $\lambda_V\mu$-term M, $\mathcal{V}term(M)$ is a finite set of proper subterms of M.*

Proof. From Proposition 3.13, M admits no infinite reduction path with respect to $\rhd_{st}$. □

Lemma 3.18. *Let M_1 and M_2 be $\lambda_V\mu$-terms. If $M_1 \rhd_{st} M_2$, then $\mathcal{VT}(M_1) \supseteq \mathcal{VT}(M_2)$.*

Proof. From Definition 3.16, we have $\mathcal{V}term(M_1) \supseteq \mathcal{V}term(M_2)$, and hence $\mathcal{VT}(M_1) \supseteq \mathcal{VT}(M_2)$. □

[5] From an analysis of Lemmata 3.3, 3.4, and 3.5, one has that if $M_1 \rhd_\mu M_2$ then $\overline{\overline{M_1}} \rhd^+_{\beta\eta} \overline{\overline{M_2}}$, provided that μ-abstractions associated with structural reductions are vacuous. This establishes the strong normalization for the full intuitionistic logic fragment of $\lambda_V\mu$, *i.e.*, including the absurdity rule.

Lemma 3.19. *If we have both* $M_1 \triangleright_{\beta\eta r} M_2$ *and* $\overline{\overline{M_1}} \not\triangleright^+_{\beta\eta} \overline{\overline{M_2}}$, *then there exist* $N_1 \in \mathcal{VT}(M_1)$ *and* $N_2 \in \mathcal{VT}(M_2)$ *such that* $N_1 \triangleright_{\beta\eta r} N_2$ *where the redex of* $\triangleright_{\beta\eta r}$ *in* M_1 *occurs in* N_1.

Proof. Let r be the $\triangleright_{\beta\eta r}$-redex in M_1, and $r \triangleright_{\beta\eta r} s$. If $\overline{\overline{M_1}}$ contained $\overline{\overline{r}}$, then $\overline{\overline{r}} \triangleright^+_{\beta\eta} \overline{\overline{s}}$, and we had $\overline{\overline{M_1}} \triangleright^+_{\beta\eta} \overline{\overline{M_2}}$. Hence, if $M_1 \triangleright_{\beta\eta r} M_2$ and $\overline{\overline{M_1}} \not\triangleright^+_{\beta\eta} \overline{\overline{M_2}}$, then the image of the $\triangleright_{\beta\eta r}$-redex in M_1 vanishes in $\overline{\overline{M_1}}$ under the modified CPS-translation. This means that an application of $\triangleright_{\beta\eta r}$ is locally executed only in a subterm of M_1, which vanishes under the modified CPS-translation. The condition $\overline{\overline{M_1}} \not\triangleright^+_{\beta\eta} \overline{\overline{M_2}}$ is caused only by the application of the case (vii) of Definition 3.8: $\overline{\overline{\mu\alpha.M}} = \overline{\overline{M}}[k := \lambda x.x][\alpha := k]$. If $\alpha \in FN(M)$, *i.e.*, $k \in FV(\overline{\overline{\mu\alpha.M}})$ for a subterm $\mu\alpha.M$ of M_1, then a term substituted into k exists in $\overline{\overline{M}}$, and hence $\triangleright_{\beta\eta r}$ is interpreted as $\triangleright^+_{\beta\eta}$. On the other hand, if $\alpha \notin FN(M)$, *i.e.*, $k \notin FV(\overline{\overline{\mu\alpha.M}})$, then substitutions into k in $\overline{\overline{M}}$ effect no change. Hence, from Definition 3.16 the set $\mathcal{V}term(M_1)$ contains every subterm of M_1 such that the modified CPS-image of the subterm is substituted into vacuous k in $\overline{\overline{\mu\alpha.M}}$ for each subterm $\mu\alpha.M$ of M_1. □

Let $\#\mathcal{VT}(M)$ denote a cardinality of the set $\mathcal{VT}(M)$.

Lemma 3.20. *Let* M_1 *and* M_2 *be* $\lambda_V\mu$*-terms. If* $M_1 \triangleright_{\beta\eta r} M_2$ *and* $\overline{\overline{M_1}} \not\triangleright^+_{\beta\eta} \overline{\overline{M_2}}$, *then* $\#\mathcal{VT}(M_1) \geq \#\mathcal{VT}(M_2)$.

Proof. We show that an application of $\triangleright_{\beta\eta r}$ adds nothing to $\mathcal{VT}(M_2)$. Assume that $M' \in \mathcal{VT}(M_2)$ or $V' \in \mathcal{VT}(M_2)$, such that the application of $\triangleright_{\beta\eta r}$ produces a new $\triangleright_{st}$-redex of the form $(\mu\alpha.M)M'$ or $V(\mu\alpha.M)$ where $\alpha \notin FN(M)$, and that $M' \notin \mathcal{V}term(M_1)$ nor $V \notin \mathcal{V}term(M_1)$. However the $\triangleright_{\beta\eta r}$-reduction is executed in some term $N \in \mathcal{V}term(M_1)$, and then N contains M' or V as a subterm. Hence, $M' \in \mathcal{VT}(M_1)$ or $V' \in \mathcal{VT}(M_1)$. □

It remains to prove that every well-typed $\lambda_V\mu$-term is strongly normalizable, including vacuous μ-abstractions. In order for the modified CPS-translation to handle the case of vacuous μ-abstractions, it is too week to prove the strong normalization that the statement: if M admits an infinite reduction sequence then so has $\overline{\overline{M}}$. Instead, we have the qualified statement: if M admits an infinite reduction sequence then there exists a subterm N of M such that $\overline{\overline{N}}$ also has an infinite reduction sequence.

Lemma 3.21. *Let* M *be a* $\lambda_V\mu$*-term. If* M *admits an infinite reduction sequence of* $\triangleright_\mu$, *then there exists a subterm* N *of* M *such that* $\overline{\overline{N}}$ *induces an infinite reduction sequence of* $\triangleright_{\beta\eta}$.

Proof. By induction on the length of M following the same case analysis as in Lemma 3.14. Case 1; σ consisting of $\triangleright_{\beta\eta r}$, can be verified following the similar pattern to Case 3 below. Case 2; σ consisting of $\triangleright_{st}$, can be proved by Proposition 3.13. Without the assumption that M has no vacuous μ-abstractions, we

show the remaining Case 3, where $\overline{\overline{M}}$ has no infinite reduction path. For simplicity, assume that the infinite reduction path σ consists of alternative $\triangleright_{\beta\eta r}$ and $\triangleright_{st}$:

$$M \triangleright_{st} M_1 \triangleright_{\beta\eta r} M_2 \triangleright_{st} M_3 \triangleright_{\beta\eta r} M_4 \triangleright_{st} M_5 \triangleright_{\beta\eta r} \cdots$$

Without loss of generality assume also the following:

$$\overline{\overline{M}} = \overline{\overline{M_1}} \triangleright^{*}_{\beta\eta} \overline{\overline{M_2}} = \overline{\overline{M_3}} \triangleright^{+}_{\beta\eta} \overline{\overline{M_4}} = \overline{\overline{M_5}} \triangleright^{*}_{\beta\eta} \cdots,$$

where $\triangleright^{*}_{\beta\eta}$ essentially means $\not\triangleright^{+}_{\beta\eta}$, *i.e.*, the reflexive closure. From the assumption that $\overline{\overline{M}}$ admits no infinite reduction path, we have infinite number of $\triangleright^{*}_{\beta\eta}$, *i.e.*, $\not\triangleright^{+}_{\beta\eta}$ except for finite number of $\triangleright^{+}_{\beta\eta}$. Otherwise, $\overline{\overline{M}}$ could induce an infinite reduction path. Then we can assume both that M_5 admits an infinite reduction path σ':

$$M_5 \triangleright_{\beta\eta r} M_6 \triangleright_{st} M_7 \triangleright_{\beta\eta r} M_8 \triangleright_{st} \cdots,$$

and that $\overline{\overline{M_5}} \triangleright^{*}_{\beta\eta} \cdots$ consists only of either "$=$" or $\not\triangleright^{+}_{\beta\eta}$, that is,

$$\overline{\overline{M_5}} = \overline{\overline{M_6}} = \overline{\overline{M_7}} = \overline{\overline{M_8}} = \cdots$$

Since every $\triangleright_{\beta\eta r}$ is interpreted as "$=$" throughout the infinite reduction sequence of $\triangleright_{\mu}$, every application of $\triangleright_{\beta\eta r}$ is locally executed only in subterms which vanish under the modified CPS-translation together with the existence of subterms $\mu\alpha.M'$ where $\alpha \notin FN(M')$, and then $\#\mathcal{VT}(M_i) > 0$. Otherwise, some of $\triangleright_{\beta\eta r}$ should have been interpreted as $\triangleright^{+}_{\beta\eta}$. Hence, from Lammata 3.18 and 3.19 we have a partial function f from a term M_i in the infinite reduction path to a term in $\mathcal{VT}(M_i)$, such that for an infinite path $M_5 \triangleright_{\beta\eta r} M_6 \triangleright_{st} M_7 \triangleright_{\beta\eta r} M_8 \triangleright_{st} \cdots$,

$$f(M_{2i-1}) = N \quad \text{for some } N \in \mathcal{VT}(M_{2i-1}) \text{ such that } N \triangleright_{\beta\eta r} N' \in \mathcal{VT}(M_{2i});$$
$$f(M_{2i}) = \begin{cases} N & \text{for some } N \in \mathcal{VT}(M_{2i}) \text{ such that } N \triangleright_{st} N' \in \mathcal{VT}(M_{2i+1}), \\ \bot & \text{otherwise.} \end{cases}$$

Then there exists a proper subterm $N \in \mathcal{VT}(M_5)$ such that N admits an infinite reduction path of $\triangleright_{\mu}$. Because the infinite reduction path σ' has infinite applications of $\triangleright_{\beta\eta r}$, while each $\mathcal{VT}(M_i)$ is a finite set such that $\#\mathcal{VT}(M_i) \geq \#\mathcal{VT}(M_{i+1}) > 0$ by Lemmata 3.17, 3.18, and 3.20. Therefore, from the induction hypothesis we have that $\overline{\overline{N}}$ induces an infinite reduction path of $\triangleright_{\beta\eta}$. □

Theorem 3.22 (*Strong Normalization Property for $\lambda_V\mu$*). *Any well-typed $\lambda_V\mu$-term is strongly normalizable.*

Proof. From Lemma 3.21 together with the strong normalization of well-typed $\lambda 2$-terms in domain-free style [7]. □

It is observed [17] that the straightforward use of the Tait and Martin–Löf parallel reduction [56] could not work for proving the Church–Rosser property for $\lambda\mu$ including renaming rule, contrary to the comments on Theorem 2.5 in [45]. Even though one defines parallel reduction $\rightrightarrows$ as usual, we cannot establish that if $M_i \rightrightarrows N_i$ $(i = 1, 2)$, then $M_1[\alpha \Leftarrow M_2] \rightrightarrows N_1[\alpha \Leftarrow N_2]$; fact (iv) in the proof of Theorem 1 in [46].

Lemma 3.23 (*Weak Church–Rosser Property for $\lambda_V\mu$*). *If $M \rhd_\mu M_1$ and $M \rhd_\mu M_2$, then $M_1 \rhd_\mu^* N$ and $M_2 \rhd_\mu^* N$ for some N.*

From Theorem 3.22 and Lemma 3.23, we can obtain the Church–Rosser property using Newman's lemma [5].

Theorem 3.24 (*Church–Rosser Theorem*). *$\lambda_V\mu$ has the Church–Rosser property for well-typed terms.*

It is to be noted that an application of parallel reductions is also studied to prove the Church–Rosser property for type-free $\lambda\mu$-calculi of both call-by-name and call-by-value [9, 10]. From the method [10], we can establish that type free $\lambda_V\mu$-calculus also enjoys the Church–Rosser property.

4. Static properties of $\lambda_V\mu$-calculus

In this section, we briefly study the problems of type checking, typability, and type inference for $\lambda_V\mu$-calculus. The problems for $\lambda_V\mu$ can be solved by answers to the corresponding problems for domain-free $\lambda 2$ that is a subsystem of $\lambda_V\mu$.

With respect to $\lambda_V\mu$, the problem of type inference is, given a term M, to decide if there exist contexts Γ, Δ and a type A such that $\Gamma \vdash M : A; \Delta$ holds in $\lambda_V\mu$, whose problem is denoted by $? \vdash M :?; ?$. On the one hand, the problem of strong type inference [57] is, given a term M and a context Γ_0, to decide if there exist contexts $\Gamma \supseteq \Gamma_0$, Δ and a type A such that $\Gamma \vdash M : A; \Delta$ is derivable. Given a term M and contexts Γ and Δ, then the typability problem is to decide if there exists a type A such that $\Gamma \vdash M : A; \Delta$ is derivable, denoted by $\Gamma \vdash M :?; \Delta$. Finally, the type checking problem is, given a term M, a type A, and contexts Γ and Δ, to decide if the judgement $\Gamma \vdash M : A; \Delta$ is derivable, denoted by $\Gamma \vdash M : A; \Delta?$.

The type inference problem is proved undecidable for domain-free $\lambda 2$ [8]. Even if the given term is in a normal form, the strong type inference is undecidable for domain-free $\lambda 2$ [20]. Both results are obtained directly or indirectly based on the undecidable second-order unification problem of Schubert [53, 54]. A well-formed expression T of the second-order unification is defined as follows:

Definition 4.1 (*Well-Formed Expressions T of Second-Order Unification*).

(1) A type variable t is a well-formed expression of a constant.
(2) If X is an n-arity variable $(n \geq 0)$ and τ_i $(1 \leq i \leq n)$ are monotypes in terms of ML, *i.e.*, types without $\forall$, then $X\tau_1 \ldots \tau_n$ is well-formed.
(3) If T_1 and T_2 are well-formed, then so is $T_1 \to T_2$.

Schubert [53,54] has proved that the halting problem for two-counter automata is reduced to the unification problem on the well-formed expressions, called simple instances of the second-order unification. Remarked that a two-counter automaton can simulate an arbitrary Turing machine [15,32,41].

Theorem 4.2 (*Schubert* [53,54]). *The second-order unification problem on the well-formed expressions is undecidable.*

From the theorem above, we directly prove that the problem of strong type inference for domain-free $\lambda 2$ is undecidable. To show this, we demonstrate a stronger result such that the problem of strong type inference is undecidable for the predicative fragment of domain-free $\lambda 2$, called domain-free ML.

Strictly speaking, the following system, domain-free ML, is a subsystem of the so-called ML, however such a subsystem is enough to establish the undecidability:

Definition 4.3 (*Domain-Free ML*).

- *Monotypes*
  $\tau ::= t \mid \tau \to \tau.$
- *Polytypes*
  $\sigma ::= \tau \mid \forall t.\sigma.$
- *Type Assumptions*
  $\Gamma ::= \langle\,\rangle \mid x{:}\sigma, \Gamma.$
- *Terms*
  $M ::= x \mid \lambda x.M \mid MM \mid x[\tau_1]\cdots[\tau_n].$
- *Type Assignment Rules*

$$\frac{\Gamma(x) = \forall t_1 \ldots t_n.\tau}{\Gamma \vdash x[\tau_1]\cdots[\tau_n] : \tau[t_1 := \tau_1, \ldots, t_n := \tau_n]}\ (n \geq 0)$$

$$\frac{\Gamma \vdash \lambda x.M : \tau_1 \to \tau_2}{\Gamma, x{:}\tau_1 \vdash M : \tau_2} \qquad \frac{\Gamma \vdash M_1 M_2 : \tau_2}{\Gamma \vdash M_1 : \tau_1 \to \tau_2 \Gamma \vdash M_2 : \tau_1}.$$

Proposition 4.4 (*Reduction from Unification to Strong Type Inference*). *The unification problem on the well-formed expressions is reduced to the problem of strong type inference for domain-free* ML. *That is, given well-formed expressions T_1 and T_2, then*

$$S(T_1) =_\beta S(T_2) \text{ under a unifier } S$$
$$\iff \exists\Gamma.\exists\tau.\ \Gamma, \Gamma_0^{T_{1,2}} \vdash M^{T_{1,2}} : \tau \text{ in domain-free ML}.$$

Outline of Proof. The context $\Gamma_0^{T_{1,2}}$ and the term $M^{T_{1,2}}$ of the strong type inference problem are determined by the given expressions T_1 and T_2. The existence of a unifier S for the unification problem gives Γ and τ, respectively. See also [20,21] for the detailed encodings. □

Proposition 4.5 (*Strong Type Inference for Domain-Free* ML). *The problem of strong type inference is undecidable for domain-free* ML.

Proof. From Theorem 4.2 and Proposition 4.4. □

Hence, the problem of strong type inference becomes undecidable for $\lambda_V\mu$ either. In the case of the domain-free style, the (strong) type inference problem for domain-free $\lambda 2$ is reduced to the typability problem, and moreover, the typability problem for domain-free $\lambda 2$ is reduced to the type checking problem, as follows:

Lemma 4.6. $\exists\Gamma.\exists A.\ \Gamma, \Gamma_0 \vdash M : A$ *in domain-free* $\lambda 2$
$\iff \exists A.\ \Gamma_0 \vdash \lambda\vec{x}.M : A$ *in domain-free* $\lambda 2$
$\iff \Gamma_0 \vdash (\lambda x.\lambda y.y)(\lambda\vec{x}.M) : t \to t$ *in domain-free* $\lambda 2$

From the undecidability of (strong) type inference for domain-free $\lambda 2$, both problems of typability and type checking become undecidable for domain-free $\lambda 2$ [20,21].

Theorem 4.7 (*Static Properties for Domain-Free* $\lambda 2$). *All of type checking, typability, and strong type inference are undecidable for domain-free* $\lambda 2$.

Proof. From Proposition 4.5 and Lemma 4.6. □

Therefore, the corresponding problems for $\lambda_V\mu$-calculus are, in general, undecidable as shown in Table 2:

TABLE 2. Decidability of type checking, typability, and type inference for $\lambda_V\mu$.

	$\Gamma \vdash M : A; \Delta?$	$\Gamma \vdash M :?; \Delta$	$? \vdash M :?; ?$
$\lambda_V\mu$-Calculus	no	no	no

Noted that the type checking problem for domain-free $\lambda 2$ becomes decidable [7,8], if the given term is a β-normal form. In the case of the call-by-name variant of domain-free $\lambda_V\mu$, the type checking becomes decidable under the same restriction. However, we do not know whether the same statement holds for the call-by-value system $\lambda_V\mu$. We say that a $\lambda_V\mu$-term M is in a restricted form if for each subterm in the form of $(M_1 \cdots M_n)$ in M $(n \geq 2)$, the head term M_1 is neither λ- nor μ-abstraction. We only know that type checking becomes decidable for $\lambda_V\mu$-calculus provided that the given term M is in the restricted form, which can be proved by induction on M.

5. COMPARISON WITH RELATED WORK AND CONCLUDING REMARKS

We briefly compare ML [11,40] plus μ-operators ($\lambda\mu_{ml}$ see [19]) with ML plus `callcc` [25]. In ML, the class of type variables is partitioned into two subclasses, *i.e.*, the applicative and the imperative type variables. The type of `callcc` is declared with imperative type variables to guarantee the soundness of the type

inference. On the basis of the classification, the typing rule for let-expression is given such that if the let-bound expression is not a value, then generalization is allowed only for applicative type variables; otherwise generalization is possible with no restriction. There is a simple translation from the ML-programs to the $\lambda\mu_{ml}$-terms, such that the two subclasses of type variables in ML are degenerated into a single class:

$$\lceil \texttt{callcc}(M) \rceil = \mu\alpha.[\alpha](\lceil M \rceil(\lambda x.[\alpha]x));$$

$$\lceil \texttt{throw } M\ N \rceil = \mu\beta.\lceil M \rceil \lceil N \rceil \text{ where } \beta \text{ is fresh.}$$

However, there are some distinctions; according to Harper *et al.* [25], the program:

$$\texttt{let } f = \texttt{callcc}(\lambda k.\lambda x.\texttt{throw } k\ (\lambda v.x))\ \texttt{in}\ (\lambda x_1 x_2.x_2)(f\ \texttt{1})(f\ \texttt{true})$$

is not typable in ML, since $\texttt{callcc}(\lambda k.\lambda x.\texttt{throw } k\ (\lambda v.x))$ with imperative type variables is not a value, and in the case of non-value expressions, polymorphism is allowed only for expressions with applicative type variables. If it were typable with `bool`, then this program was reduced to `1` following the operational semantics. On the other hand, under the translation $\lceil\ \rceil$ together with type annotation, in explicitly typed $\lambda\mu_{ml}$ ($\lambda\mu_{eml}$ [19]) we have the following expression:

$$\texttt{let } f = \lambda t.\mu\alpha.[\alpha](\lambda x.\mu\beta.[\alpha](\lambda v.x))\ \texttt{in}\ (\lambda x_1 x_2.x_2)(f\ \texttt{int}\ \texttt{1})(f\ \texttt{bool}\ \texttt{true})$$

with type `bool`, and this expression is now reduced to `true`, as in F_ω plus `callcc` under call-by-value, not under ML-like call-by-value [27]. In turn, the following term:

$$\texttt{let } f = \mu\alpha.[\alpha](\lambda x.\mu\beta.[\alpha](\lambda v.x))\ \texttt{in}\ (\lambda x_1 x_2.x_2)(f\ \texttt{1})(f\ \texttt{2})$$

with type `int` is reduced to `1` by the symmetric structural reduction. On the other hand, in implicitly typed $\lambda\mu_{ml}$ ($\lambda\mu_{iml}$ [19]) we have the term:

$$\texttt{let } f = \mu\alpha.[\alpha](\lambda x.\mu\beta.[\alpha](\lambda v.x))\ \texttt{in}\ (\lambda x_1 x_2.x_2)(f\ \texttt{1})(f\ \texttt{true})$$

with type `bool`, and this is also reduced to `true`. $\lambda\mu_{ml}$ could overcome the counterexample of polymorphic `callcc` in ML, and moreover, the typing conditions for let-expression could be deleted. In particular, $\lambda\mu_{iml}$ is another candidate for implicit polymorphism by value, compared with implicit polymorphism by name in Leroy [36].

Ong and Stewart [45] extensively studied a call-by-value programming language based on a call-by-value variant of finitely typed $\lambda\mu$-calculus. There are some distinctions between Ong and Stewart and our finite type fragment; their reduction rules have type annotations like the complete Church-style, and, using the annotation, more reduction rules are defined than ours, which can give a stronger normal form. In addition, our notion of values is an extended one, which would be justified by observation based on the CPS-translation. Moreover, our renaming rule is applied for the extended values, and following the proof of Lemma 3.5, this point is essential for the soundness of the CPS-translation including renaming rules. Otherwise the reductions by renaming rules could not be simulated by β-reductions. In the case of μ-abstraction if $\lambda_V\mu$-terms were restricted to $\mu\alpha.[\beta]M$ instead of $\mu\alpha.M$

for any M, then one could have $\overline{[\gamma](\mu\alpha.[\beta]M)} \rhd_\beta^+ \overline{([\beta]M)[\alpha := \gamma]}$. In this sense our renaming rule for the extended values is an extended rule. On the other hand, in the equational theory λ_C of Hofmann [30], one obtains $\alpha(\mathcal{C}(\lambda\beta.M)) =_C M[\beta := \alpha]$ without restricting to values, which would be distinction between equational theory and rewriting theory.

We used the CPS-translations as a useful tool to show consistency and strong normalization of the system. With respect to Proposition 3.6 (soundness of CPS-translation); for call-by-name $\lambda\mu$, on the one hand, the completeness is obtained in de Groote [23], *i.e.*, the call-by-name CPS-translation is injective. For a call-by-value system with Felleisen's control operators [14], on the other hand, the completeness is established with respect to categorical models [30], and moreover, this method is successfully applied to call-by-name $\lambda\mu$ [31]. We believe that our CPS-translations would be natural along the line of [49], and it is worth pursuing the detailed relation to such categorical models [31, 55].

Finally, we summarize the results obtained here, besides already known ones with respect to second-order $\lambda\mu$-calculi as far as we know (Tab. 3):

TABLE 3. Properties of second-order $\lambda\mu$-calculi.

	Style	Strategy	SR	SN	CR	TYP	TC
$\lambda\mu$ [48]	Church	CBN	yes	yes	yes	yes	yes
$\lambda\mu$ [48]	Curry	CBN	yes	yes	yes	no [58]	no [58]
$\lambda_V\mu^*$	Curry	CBV	no^1	no^1	yes^2	no [58]	no [58]
$\lambda_V\mu$	Domain Free	CBV	yes^3	yes^4	yes^4	no^5	no^5

We write CBN for call-by-name, CVB for call-by-value; SR for subject reduction, SN for strong normalization, CR for Church–Rosser; and TYP for decidability of typability, TC for decidability of type checking.

All the properties of SR, SN, and CR for the $\lambda\mu$-calculus are due to Parigot [46–48]. The system $\lambda_V\mu^*$ denotes a Curry style version of $\lambda_V\mu$ proposed in this paper. Section 1 shows no^1; yes^2 is derived by the method of [10]; yes^3 is obtained in Section 2; yes^4 is in Section 3; and no^5 is in Section 4.

APPENDIX

5.1. PROOF OF LEMMA 3.3

Lemma 3.3. *Let M contain i free occurrences of $[\alpha]$ where $i \geq 0$.*

Then we have that $\overline{M[\alpha \Leftarrow M_1]} \rhd_\beta^i \overline{M}[\alpha := \lambda m.\overline{M_1}(\lambda n.mn\alpha)]$ and $\overline{M[\alpha \Leftarrow A]} \rhd_\beta^i \overline{M}[\alpha := \lambda m.mA^q\alpha]$.

Proof. By induction on the structure of M. We show only the following case:
Case of $[\alpha]M$, where M contains i free occurrences of $[\alpha]$:
$\overline{([\alpha]M)[\alpha \Leftarrow M_1]} = \lambda k.k((\lambda k'.\overline{M[\alpha \Leftarrow M_1]}\lambda m.\overline{M_1}(\lambda n.mnk'))\alpha)$

$\triangleright_\beta \lambda k.k(\overline{M[\alpha \Leftarrow M_1]}\lambda m.\overline{M_1}(\lambda n.mn\alpha))$
$\triangleright_\beta^i \lambda k.k(\overline{M}[\alpha := \lambda m.\overline{M_1}(\lambda n.mn\alpha)](\lambda m.\overline{M_1}(\lambda n.mn\alpha)))$
$= \overline{[\alpha]M}[\alpha := \lambda m.\overline{M_1}(\lambda n.mn\alpha)].$ □

5.2. Proof of Lemma 3.4

Lemma 3.4. *For any term M and value V, $\overline{M[V \Rightarrow \alpha]} \triangleright_\beta^{3i} \overline{M}[\alpha := \lambda n.\Psi(V)n\alpha]$, where M contains i free occurrences of $[\alpha]$.*

Proof. By induction on the structure of M. Only the case of $[\alpha]M$ is shown, where M contains i-occurrences of $[\alpha]$:
$\overline{([\alpha]M)[V \Rightarrow \alpha]} = \lambda k.k((\lambda k'.\overline{V}(\lambda m.\overline{M[V \Rightarrow \alpha]}(\lambda n.mnk')))\alpha)$
$\triangleright_\beta \lambda k.k((\lambda k'.k'\Psi(V))(\lambda m.\overline{M[V \Rightarrow \alpha]}(\lambda n.mn\alpha)))$
$\triangleright_\beta^2 \lambda k.k(\overline{M[V \Rightarrow \alpha]}(\lambda n.\Psi(V)n\alpha))$
$\triangleright_\beta^{3i} \lambda k.k(\overline{M}[\alpha := \lambda n.\Psi(V)n\alpha](\lambda n.\Psi(V)n\alpha))$
$= \overline{[\alpha]M}[\alpha := \lambda n.\Psi(V)n\alpha].$ □

5.3. Proof of Lemma 3.5

Lemma 3.5. *If we have $M \triangleright_\mu N$ in $\lambda_V\mu$, then $\overline{M} =_{\beta\eta} \overline{N}$ in domain-free $\lambda 2$. To be precise, the following holds true:*

(1) *if $M \triangleright_{\beta\eta r} N$ in $\lambda_V\mu$, then $\overline{M} \triangleright_{\beta\eta}^+ \overline{N}$ in domain-free $\lambda 2$;*
(2) *if $M \triangleright_{st} N$ in $\lambda_V\mu$, then $\overline{M} =_\beta \overline{N}$ in domain-free $\lambda 2$.*

Proof. By induction on the derivation of $M \triangleright_\mu N$. We show the base cases:

Case of (β_v) $(\lambda x.M)V \triangleright M[x := V]$:
$\overline{(\lambda x.M)V} = \lambda k_1.(\lambda k_2.k_2(\lambda x.\overline{M}))(\lambda m.\overline{V}(\lambda n.mnk_1))$
$\triangleright_\beta^2 \lambda k_1.\overline{V}(\lambda n.(\lambda x.\overline{M})nk_1)$
$\triangleright_\beta \lambda k_1.\overline{V}(\lambda x.\overline{M}k_1) = \lambda k_1.(\lambda k.k\Psi(V))(\lambda x.\overline{M}k_1)$
$\triangleright_\beta^2 \lambda k_1.\overline{M}[x := \Psi(V)]k_1 = \lambda k_1.\overline{M[x := V]}k_1 \triangleright_\beta \overline{M[x := V]}.$

Case of (η_v) $\lambda x.Vx \triangleright V$ where $x \notin FV(V)$:
$\overline{\lambda x.Vx} = \lambda k.k(\lambda x.(\lambda k'.(\overline{V}(\lambda m.\overline{x}(\lambda n.mnk')))))$
$\triangleright_\beta^2 \lambda k.k(\lambda x.(\lambda k'.(\overline{V}(\lambda m.mxk')))) = \lambda k.k(\lambda x.(\lambda k'.(\lambda k''.k''\Psi(V))(\lambda m.mxk')))$
$\triangleright_\beta^2 \lambda k.k(\lambda x.(\lambda k'.\Psi(V)xk'))$
$\triangleright_\eta \lambda k.k(\lambda x.\Psi(V)x) \triangleright_\eta \lambda k.k\Psi(V) = \overline{V}.$

Case of (β_t) $(\lambda t.M)A \triangleright M$:
$\overline{(\lambda t.M)A} = \lambda k.(\lambda k'.k'(\lambda t.\overline{M}))(\lambda m.mA^qk)$
$\triangleright_\beta^2 \lambda k.(\lambda t.\overline{M})A^qk \triangleright_\beta \lambda k.\overline{M}[t := A^q]k \triangleright_\beta \overline{M[t := A]}.$

Case of (η_t) $\lambda t.Vt \triangleright V$ where $t \notin FV(V)$:
$\overline{\lambda t.Vt} = \lambda k.k(\lambda t.(\lambda k.\overline{V}(\lambda m.mtk)))$
$= \lambda k.k(\lambda t.(\lambda k.(\lambda k.k\Psi(V))(\lambda m.mtk)))$
$\triangleright_\beta^2 \ \lambda k.k(\lambda t.(\lambda k.\Psi(V)tk))$
$\triangleright_\eta^2 \ \lambda k.k\Psi(V) = \overline{V}$.

Case of (μ_t) $(\mu\alpha.M)A \triangleright \mu\alpha.M[\alpha \Leftarrow A]$:
$\overline{(\mu\alpha.M)A} = \lambda k.(\lambda\alpha.\overline{M}(\lambda x.x))(\lambda m.mA^q k)$
$\triangleright_\beta \lambda\alpha.\overline{M}[\alpha := \lambda m.mA^q\alpha](\lambda x.x) =_\beta \lambda\alpha.\overline{M[\alpha \Leftarrow A]}(\lambda x.x) = \overline{\mu\alpha.M[\alpha \Leftarrow A]}$.

Case of (μ_l) $(\mu\alpha.M)N \triangleright \mu\alpha.M[\alpha \Leftarrow N]$:
$\overline{(\mu\alpha.M)N} = \lambda k.(\lambda\alpha.\overline{M}(\lambda x.x))(\lambda m.\overline{N}(\lambda n.mnk))$
$\triangleright_\beta \lambda k.\overline{M}[\alpha := \lambda m.\overline{N}(\lambda n.mnk)](\lambda x.x) = \lambda\alpha.\overline{M}[\alpha := \lambda m.\overline{N}(\lambda n.mn\alpha)](\lambda x.x)$
$=_\beta \ \lambda\alpha.\overline{M[\alpha \Leftarrow N]}(\lambda x.x) = \overline{\mu\alpha.M[\alpha \Leftarrow N]}$.

Case of (μ_r) $V(\mu\alpha M) \triangleright \mu\alpha.M[V \Rightarrow \alpha]$:
$\overline{V(\mu\alpha.M)} = \lambda k.\overline{V}(\lambda m.(\lambda\alpha.\overline{M}(\lambda x.x))(\lambda n.mnk))$
$\triangleright_\beta \lambda k.(\lambda k'.k'\Psi(V))(\lambda m.\overline{M}[\alpha := \lambda n.mnk](\lambda x.x))$
$\triangleright_\beta^2 \lambda k.\overline{M}[\alpha := \lambda n.\Psi(V)nk](\lambda x.x) = \lambda\alpha.\overline{M}[\alpha := \lambda n.\Psi(V)n\alpha](\lambda x.x)$
$=_\beta \ \lambda\alpha.\overline{M[V \Rightarrow \alpha]}(\lambda x.x) = \overline{\mu\alpha.M[V \Rightarrow \alpha]}$.

Case of (rn) $[\alpha](\mu\beta.V) \triangleright V[\beta := \alpha]$:
$\overline{[\alpha](\mu\beta.V)} = \lambda k.k((\lambda\beta.\overline{V}(\lambda x.x))\alpha)$
$\triangleright_\beta \lambda k.k(\overline{V}[\beta := \alpha](\lambda x.x)) = \lambda k.k((\lambda k'.k'\Psi(V)[\beta := \alpha])(\lambda x.x))$
$\triangleright_\beta^2 \lambda k.k\Psi(V)[\beta := \alpha] = \overline{V[\beta := \alpha]}$.

Case of $(\mu\text{-}\eta)$ $\mu\alpha.[\alpha]M \triangleright M$ where $\alpha \notin FN(M)$:
$\overline{\mu\alpha.[\alpha]M} = \lambda\alpha.(\lambda k.k(\overline{M}\alpha))(\lambda x.x)$
$\triangleright_\beta^2 \lambda\alpha.\overline{M}\alpha \triangleright_\beta \overline{M}$. □

5.4. Proof of Lemma 3.9

Lemma 3.9. *We have the following property with respect to replacements:*

(1) $\overline{\overline{M[\alpha \Leftarrow N]}} = \overline{\overline{M}[\alpha := \lambda m.\overline{\overline{N}}[k := \lambda n.mn\alpha]]}$.
(2) $\overline{\overline{M[\alpha \Leftarrow A]}} = \overline{\overline{M}[\alpha := \lambda m.mA^q\alpha]}$.
(3) $\overline{\overline{M[V \Rightarrow \alpha]}} = \overline{\overline{M}[\alpha := \lambda n.\Phi(V)n\alpha]}$.

Proof. By induction on the structure of M. We show one case M of $[\alpha]M'$, where $=_{ih}$ denotes the use of the induction hypothesis:

(1) $\overline{\overline{([\alpha]M)[\alpha \Leftarrow N]}} = \overline{\overline{[\alpha](M[\alpha \Leftarrow N])N}}$
$= k(\overline{\overline{M[\alpha \Leftarrow N]N}}[k := \alpha])$

$= k(\overline{\overline{M[\alpha \Leftarrow N]}}[k := \lambda m.\overline{\overline{N}}[k := \lambda n.mnk]][k := \alpha])$
$=_{ih} k(\overline{\overline{M}}[\alpha := \lambda n.\overline{\overline{N}}[k := \lambda n.mn\alpha]][k := \lambda m.\overline{\overline{N}}[k := \lambda n.mnk]][k := \alpha])$
$= k(\overline{\overline{M}}[k := \alpha][\alpha := \lambda m.\overline{\overline{N}}[k := \lambda n.mn\alpha]])$
$= \overline{\overline{[\alpha]M}}[\alpha := \lambda m.\overline{\overline{N}}[k := \lambda n.mn\alpha]].$

(2) $\overline{\overline{([\alpha]M)[\alpha \Leftarrow A]}} = \overline{\overline{[\alpha](M[\alpha \Leftarrow A])A}}$
$= k(\overline{\overline{(M[\alpha \Leftarrow A])A}}[k := \alpha])$
$= k(\overline{\overline{M[\alpha \Leftarrow A]}}[k := \lambda m.mA^q k][k := \alpha])$
$=_{ih} k(\overline{\overline{M}}[\alpha := \lambda m.mA^q\alpha][k := \lambda m.mA^q\alpha])$
$= k(\overline{\overline{M}}[k := \alpha])[\alpha := \lambda m.mA^q\alpha]$
$= \overline{\overline{[\alpha]M}}[\alpha := \lambda m.mA^q\alpha].$

(3) $\overline{\overline{[\alpha](M[V \Rightarrow \alpha])}} = \overline{\overline{[\alpha]V(M[V \Rightarrow \alpha])}}$
$= k(\overline{\overline{V(M[V \Rightarrow \alpha])}}[k := \alpha])$
$= k(\overline{\overline{M[V \Rightarrow \alpha]}}[k := \lambda n.\Phi(V)nk][k := \alpha])$
$=_{ih} k(\overline{\overline{M}}[\alpha := \lambda n.\Phi(V)n\alpha][k := \lambda n.\Phi(V)nk][k := \alpha])$
$= k(\overline{\overline{M}}[\alpha := \lambda n.\Phi(V)n\alpha][k := \lambda n.\Phi(V)n\alpha])$
$= k(\overline{\overline{M}}[k := \alpha][\alpha := \lambda n.\Phi(V)n\alpha])$
$= \overline{\overline{[\alpha]M}}[\alpha := \lambda n.\Phi(V)n\alpha].$ □

5.5. Proof of Lemma 3.10

Lemma 3.10. *If we have* $M_1 \rhd_\mu M_2$ *in* $\lambda_V\mu$, *then* $\overline{\overline{M_1}} \rhd^*_{\beta\eta} \overline{\overline{M_2}}$ *in domain-free* $\lambda 2$. *To be precise, the following holds true:*

(1) *if* $M_1 \rhd_{\beta\eta r} M_2$ *in* $\lambda_V\mu$, *then* $\overline{\overline{M_1}} \rhd^*_{\beta\eta} \overline{\overline{M_2}}$ *in domain-free* $\lambda 2$.
More precisely, let M_1 *contain no vacuous* μ*-abstraction, i.e.,* $\alpha \in FN(M)$ *for every subterm* $\mu\alpha.M$ *of* M_1. *If* $M_1 \rhd_{\beta\eta r} M_2$, *then* $\overline{\overline{M_1}} \rhd^+_{\beta\eta} \overline{\overline{M_2}}$ *in domain-free* $\lambda 2$;

(2) *if* $M_1 \rhd_{st} M_2$ *in* $\lambda_V\mu$, *then* $\overline{\overline{M_1}} = \overline{\overline{M_2}}$ *in domain-free* $\lambda 2$.

Proof. By induction on the derivation of $M_1 \rhd_\mu M_2$. For the case of (1), see also the proof of Lemma 3.19. We show the base cases below, where id denotes $\lambda x.x$:

Case of (β_v) $(\lambda x.M)V \rhd M[x := V]$:
$\overline{\overline{(\lambda x.M)V}} = \Phi(\lambda x.M)\Phi(V)k = (\lambda x.\lambda k.\overline{\overline{M}})\Phi(V)k$
$\rhd_\beta \lambda k.\overline{\overline{M}}[x := \Phi(V)]k \rhd_\eta \overline{\overline{M}}[x := \Phi(V)] = \overline{\overline{M[x := V]}}.$

Case of (η_v) $\lambda x.Vx \rhd V$ where $x \notin FV(V)$:
$\overline{\overline{\lambda x.Vx}} = k(\lambda x.\lambda k.\overline{\overline{Vx}}) = k(\lambda x.\lambda k.\Phi(V)\Phi(x)k)$

$\triangleright_\eta\ k(\lambda x.\Phi(V)x) \triangleright_\eta\ k\Phi(V) = \overline{\overline{V}}$.

Case of (β_t) $(\lambda t.M)A \triangleright M[t := A]$:
$\overline{\overline{(\lambda t.M)A}} = \Phi(\lambda t.M)A^q k = (\lambda t.\lambda k.\overline{\overline{M}})A^q k$
$\triangleright_{\beta_t}\ \lambda k.\overline{\overline{M}}[t := A^q]k \triangleright_\eta\ \overline{\overline{M[t := A]}}$.

Case of (η_t) $\lambda t.Vt \triangleright V$ where $t \notin FV(V)$:
$\overline{\overline{\lambda t.Vt}} = k(\lambda t.\lambda k.\Phi(V)tk)$
$\triangleright_\eta^2\ k\Phi(V) = \overline{\overline{V}}$.

Case of (μ_t) $(\mu\alpha.M)A \triangleright \mu\alpha.M[\alpha \Leftarrow A]$:
$\overline{\overline{(\mu\alpha.M)A}} = \overline{\overline{\mu\alpha.M}}[k := \lambda m.mA^q k]$
$= \overline{\overline{M}}[k := id][\alpha := k][k := \lambda m.mA^q k]$
$= \overline{\overline{M}}[\alpha := \lambda m.mA^q\alpha][k := id][\alpha := k]$
$= \overline{\overline{M[\alpha \Leftarrow A]}}[k := id][\alpha := k] = \overline{\overline{\mu\alpha.M[\alpha \Leftarrow A]}}$.

Case of (μ_l) $(\mu\alpha.M)N \triangleright \mu\alpha.M[\alpha \Leftarrow N]$:
$\overline{\overline{(\mu\alpha.M)N}} = \overline{\overline{M}}[k := id][\alpha := k][k := \lambda m.\overline{\overline{N}}[k := \lambda n.mnk]]$
$= \overline{\overline{M}}[k := id][\alpha := \lambda m.\overline{\overline{N}}[k := \lambda n.mnk]]$
$= \overline{\overline{M}}[\alpha := \lambda m.\overline{\overline{N}}[k := \lambda n.mn\alpha]][k := id][\alpha := k]$
$= \overline{\overline{M[\alpha \Leftarrow N]}}[k := id][\alpha := k] = \overline{\overline{\mu\alpha.M[\alpha \Leftarrow N]}}$.

Case of (μ_r) $V(\mu\alpha.M) \triangleright \mu\alpha.M[V \Leftarrow \alpha]$:
$\overline{\overline{V(\mu\alpha.M)}} = \overline{\overline{\mu\alpha.M}}[k := \lambda n.\Phi(V)nk]$
$= \overline{\overline{M}}[k := id][\alpha := k][k := \lambda n.\Phi(V)nk]$
$= \overline{\overline{M}}[\alpha := \lambda n.\Phi(V)n\alpha][k := id][\alpha := k]$
$= \overline{\overline{M[V \Rightarrow \alpha]}}[k := id][\alpha := k] = \overline{\overline{\mu\alpha.M[V \Rightarrow \alpha]}}$.

Case of $(r\eta)$ $[\alpha](\mu\beta.V) \triangleright V[\beta := \alpha]$:
$\overline{\overline{[\alpha](\mu\beta.V)}} = k(\overline{\overline{\mu\beta.V}}[k := \alpha])$
$= k(\overline{\overline{V}}[k := id][\beta := k][k := \alpha])$
$= k(\overline{\overline{V}}[k := id][\beta := \alpha])$
$\triangleright_\beta\ k(\Phi(V)[\beta := \alpha]) = \overline{\overline{V[\beta := \alpha]}}$.

Case of (μ_η) $\mu\alpha.[\alpha]M \triangleright M$ where $\alpha \notin FN(M)$:
$\overline{\overline{\mu\alpha.[\alpha]M}} = \overline{\overline{[\alpha]M}}[k := id][\alpha := k]$
$= (k(\overline{\overline{M}}[k := \alpha]))[k := id][\alpha := k]$
$\triangleright_\beta\ \overline{\overline{M}}[k := \alpha][\alpha := k] = \overline{\overline{M}}$. □

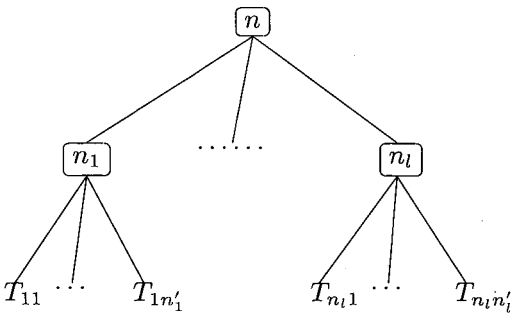
5.6. Proof of Proposition 3.13

Proposition 3.13 (*Strong Normalization of* $\triangleright_{st}$). *Any* $\lambda_V\mu$*-term is strongly normalizable with respect to the structural reductions* $\triangleright_{st}$.

Proof. From Lemma 5.7 below. □

Recall that a term $\mathcal{E}[\mu\alpha.M]$ is also represented as $\mathcal{E}_n[\mathcal{E}_{n-1}[\cdots\mathcal{E}_1[\mu\alpha.M]\cdots]]$, where $\mathcal{E}_i[\ ]$ is either $[\ ]M_i$, $V_i[\ ]$, or $[\ ]A_i$, and for simplicity, $\mathcal{E}_i$ also denotes the terms M_i, V_i, or the type A_i.

The term $\mathcal{E}_n[\mathcal{E}_{n-1}[\cdots\mathcal{E}_1[\mu\alpha.M]\cdots]]$ where $\mathcal{E}_i \not\equiv [\ ]$ has successive redexes of $\triangleright_{st}$, and the size of the redexes will be defned by the natural number n. In the case $\mathcal{E}$ of $[\ ]$, we say that the size of redexes of $\mathcal{E}[\mu\alpha.M]$ is 0. Since a term-tree is finite, the structure of redexes will be represented as a finite tree whose nodes are labelled with the size (a natural number), especially leaves with 0:

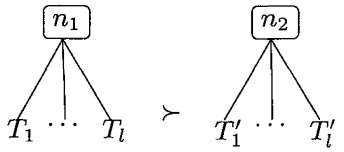


where the root node is denoted by the natural number $\boxed{n}$ and the children are represented by $\boxed{n_1}$, ..., $\boxed{n_l}$. The branching number l in this example will be determined by l free occurrences of $[\alpha]$ in M of $\mathcal{E}_n[\mathcal{E}_{n-1}[\cdots\mathcal{E}_1[\mu\alpha.M]\cdots]]$.

An ordering on the trees T is defined as follows:

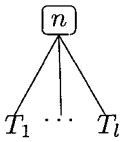
Definition 5.1 (*Ordering on Trees,* $\succ$).

(I) A tree whose root is smaller is smaller:



if $n_1 > n_2$.

(II) A tree whose subtree is removed is smaller:
Let T be the following tree with root n:



Then $T \succ T'$, where T' is obtained from T by removing some T_i for $1 \leq i \leq l$.

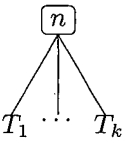
(III) A tree whose subtree is smaller is smaller:
If $T[T_1']$ is a tree obtained by repalcing a subtree T_1 of T with T_1' such that $T_1 \succ T_1'$, then $T[T_1] \succ T[T_1']$.

The ordering $\succ$ on the trees is well-founded, since we have the minimum tree 0.

Given a $\lambda_V\mu$-term, then we first define a degree of the term with respect to successive redexes of $\triangleright_{st}$, that is, $\mathcal{E}_n[\mathcal{E}_{n-1}[\cdots\mathcal{E}_1[\mu\alpha.M]\cdots]]$:

Definition 5.2 (*degree*, $\deg(\mathcal{E}(\mu\alpha.M))$).

$$\deg(\mathcal{E}_n[\mathcal{E}_{n-1}[\cdots\mathcal{E}_1[\mu\alpha.M_1]\cdots]]) \quad = \quad$$



where $n \geq 0$, and the branching number k is obtained by k free occurrences of $[\alpha]$ in M_1. For $1 \leq i \leq k$, the subtree T_i is defined by the shape of the subterm $[\alpha]M'$ of M_1, as follows:

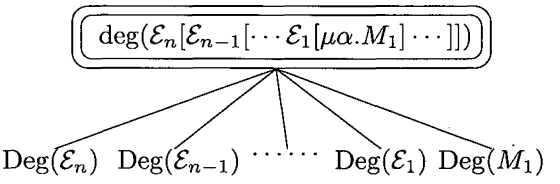
$$T_i = \begin{cases} \deg(\mathcal{E}[\mu\beta.M'']) & \text{for a subterm } [\alpha](\mathcal{E}[\mu\beta.M'']) \text{ of } M_1; \\ \deg(\mathcal{E}_1) & \text{for a subterm } [\alpha]V \text{ and } \mathcal{E}_1 \equiv \mathcal{E}[\mu\beta.M'']; \\ 0 & \text{otherwise.} \end{cases}$$

The depth of the tree can be regarded as a length of the segment in terms of $[1,50]$.

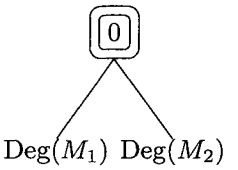
Next we define a tree of the trees, denoted by $\mathcal{T}$ or $\mathrm{Deg}(M)$, which means a degree of the whole term M. A node of a tree is denoted by $\boxed{n}$, and a node of a tree of trees is by $\boxed{\boxed{T}}$. Given a $\lambda_V\mu$-term M, then $\mathrm{Deg}(M)$ is defined in the following:

Definition 5.3 (*Degree*, $\mathrm{Deg}(M)$).

(1) Case M of $\mathcal{E}_n[\mathcal{E}_{n-1}[\cdots\mathcal{E}_1[\mu\alpha.M_1]\cdots]]$ where $n \geq 0$:
$\mathrm{Deg}(M)$ is a tree of trees, consisting of $\deg(\mathcal{E}_n[\mathcal{E}_{n-1}[\cdots\mathcal{E}_1[\mu\alpha.M_1]\cdots]])$ as a root and consisting of $\mathrm{Deg}(\mathcal{E}_1), \ldots, \mathrm{Deg}(\mathcal{E}_n)$, $\mathrm{Deg}(M_1)$ as the subtrees, as follows:

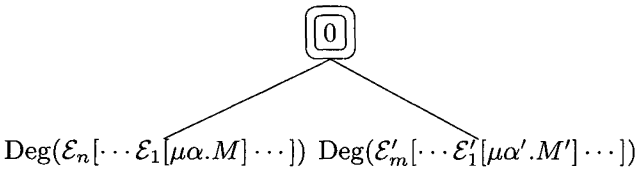


(2) Other case M of term application; $M \equiv M_1M_2$:



(3) Otherwise:
$\mathrm{Deg}(\lambda x.M) = \mathrm{Deg}(\lambda t.M) = \mathrm{Deg}([\alpha]M) = \mathrm{Deg}(MA) = \mathrm{Deg}(M)$;
$\mathrm{Deg}(x) = 0$.

When a $\lambda_V\mu$-term is, for instance, in the form of $(\lambda x.\mathcal{E}_n[\cdots\mathcal{E}_1[\mu\alpha.M]\cdots])\ (\lambda y.\mathcal{E}'_m[\cdots\mathcal{E}'_1[\mu\alpha'.M']\cdots])$, Deg of the term is defined as follows:

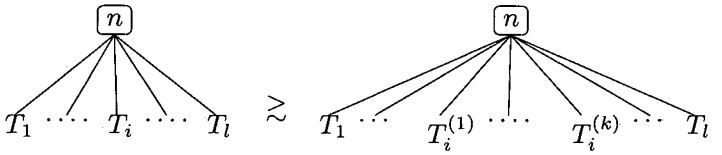


In order to give an ordering on the trees of trees $\mathcal{T}$, we give an auxiliary ordering $\gtrsim$ on trees T:

Definition 5.4 (*Auxiliary Relation on Trees*, $\gtrsim$).

(i) $T \gtrsim T$

(ii)



where $l \geq 1$ and $1 \leq i \leq l$. Here, the right-hand side is obtained from the left-hand side by k times copying a subtree T_i ($k \geq 1$), and the subtree T_i is called a copied subtree.

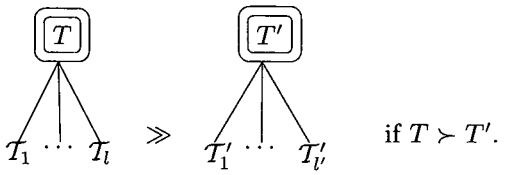
(iii) If $T_1 \gtrsim T_2$, then $T[T_1] \gtrsim T[T_2]$.

From the definition, if $T_1 \gtrsim T_2$ then either $T_1 \equiv T_2$ or T_2 contains a copied subtree. It is clear that the relation $\gtrsim$ is not well-founded.

A tree is denoted by T and a tree of trees is by $\mathcal{T}$. The orderings on trees are extended to the trees of trees, denoted by $\gg$.

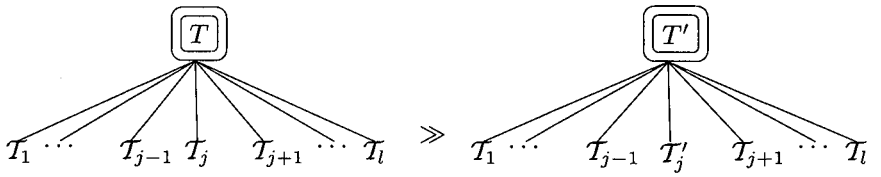
Definition 5.5 (*Ordering on Trees of Trees*, $\gg$).

(IV) A tree of trees whose root is smaller is smaller:



if $T \succ T'$.

(V) A tree of trees whose subtree is smaller is smaller, where in the root a subtree can be copied:
If $\mathcal{T}_j \gg \mathcal{T}'_j$ and $T \gtrsim T'$, then



(VI) A tree of trees whose subtree is smaller is smaller:
If $\mathcal{T}[\mathcal{T}'_1]$ is a tree of trees obtained by repalcing a subtree $\mathcal{T}_1$ of $\mathcal{T}$ with $\mathcal{T}'_1$ such that $\mathcal{T}_1 \gg \mathcal{T}'_1$, then $\mathcal{T}[\mathcal{T}_1] \gg \mathcal{T}[\mathcal{T}'_1]$.

The ordering $\gg$ on the trees of trees is also well-founded, since we have a minimal tree in which every component tree is 0, *i.e.* $\boxed{\boxed{0}}$.

For natural number $n \geq 1$, the $\lambda_V\mu$-term P_n that has successive redexes of structural reductions is defined as follows:

$$\begin{cases} P_1 = \mathcal{E}^1[\mu\alpha_1.P_0] & \text{where } P_0 \text{ contains no subterm of the form } [\alpha_1](\mathcal{E}^0[\mu\alpha_0.M']); \\ P_{n+1} = \mathcal{E}^{n+1}[\mu\alpha_{n+1}.N_{n+1}] & \text{where } N_{n+1} \text{ contains a subterm of the form } [\alpha_{n+1}]P_k \text{ for some } k \leq n. \end{cases}$$

Let $[X/\alpha]$ be either $[\alpha \Leftarrow X]$ or $[X \Rightarrow \alpha]$.

Lemma 5.6. *Let M_1 be $\mathcal{E}_n[\cdots\mathcal{E}_1[\mu\alpha.M]\cdots]$ where $n \geq 0$.*
If $M_1 \rhd_{st} M_2$, then either $\deg(M_1) \succ \deg(M_2)$ or $\deg(M_1) \gtrsim \deg(M_2)$.

Proof. By induction on the length of M_1, we show that either $\deg(M_1) \succ \deg(M_2)$ or $\deg(M_1) \gtrsim \deg(M_2)$, if $M_1 \rhd_{st} M_2$.

Case 1. $M_1 \equiv \mathcal{E}_n[\cdots\mathcal{E}_1[\mu\alpha.M]\cdots] \rhd_{st} \mathcal{E}_n[\cdot\,\mathcal{E}_2[\mu\alpha.M[\mathcal{E}_1/\alpha]]\,\cdot] \equiv M_2$: if $\mathcal{E}_n[\cdots\mathcal{E}_1[\mu\alpha.M]\cdots] \rhd_{st} \mathcal{E}_n[\cdot\,\mathcal{E}_2[\mu\alpha.M[\mathcal{E}_1/\alpha]]\,\cdot]$, then from (I) of Definition 5.1, one has $\deg(\mathcal{E}_n[\cdots\mathcal{E}_1[\mu\alpha.M]\cdots]) \succ \deg(\mathcal{E}_n[\cdot\,\mathcal{E}_2[\mu\alpha.M[\mathcal{E}_1/\alpha]]\,\cdot])$.

Case 2. $M_1 \equiv \mathcal{E}_n[\cdots\mathcal{E}_1[\mu\alpha.M]\cdots] \rhd_{st} \mathcal{E}_n[\cdots\mathcal{E}_1[\mu\alpha.M']\cdots] \equiv M_2$, where $M \rhd_{st} M'$: we have the two cases where successive redexes P_k in M is reduced or not.

Case 2-1. Successive redexes P_k in M is reduced: if $M_1 \equiv P_{m+1} \equiv \mathcal{E}^{m+1}[\mu\alpha_{m+1}.N_{m+1}] \rhd_{st} \mathcal{E}^{m+1}[\mu\alpha_{m+1}.N'_{m+1}] \equiv M_2$, where N'_{m+1} is obtained from N_{m+1} by reducing a subterm $P_{k\leq m}$ of N_{m+1} in the same way as Case 1 above, then

$\deg(\mathcal{E}^{m+1}[\mu\alpha_{m+1}.N_{m+1}]) \succ \deg(\mathcal{E}^{m+1}[\mu\alpha_{m+1}.N'_{m+1}])$ by (I) and (III) of Definition 5.1.

Case 2-2. Non-successive redexes $\mathcal{E}'[\mu\beta.M_3]$ is reduced in M, where $\mathcal{E}'[\mu\beta.M_3] \rhd_{st} \mu\beta.M_3[\mathcal{E}'/\beta]$:

Case 2-2-1. $\mathcal{E}'$ of $\mathcal{E}'[\mu\beta.M_3]$ in M contains successive redexes P_k:

Case 2-2-1-1. $\beta \notin FN(M_3)$: we have $\mathcal{E}'[\mu\beta.M_3] \rhd_{st} \mu\beta.M_3$, and hence $\deg(M_1) \succ \deg(M_2)$ from (II) and (III) of Definition 5.1.

Case 2-2-1-2. $\beta \in FN(M_3)$: after the reduction $\mathcal{E}'[\mu\beta.M_3] \rhd_{st} \mu\beta.M_3[\mathcal{E}'/\beta]$, the successive redexes P_k in $\mathcal{E}'$ is copied as the number of the free occurrences of β in M_3. Hence, $\deg(M_1) \gtrsim \deg(M_2)$ by Definition 5.4.

Case 2-2-2. $\mathcal{E}'$ of $\mathcal{E}'[\mu\beta.M_3]$ in M contains no successive redexes: even after the reduction $\mathcal{E}'[\mu\beta.M_3] \rhd_{st} \mu\beta.M_3[\mathcal{E}'/\beta]$, $\deg(M_1)$ has no change, since $\mathcal{E}'$ contains no P_k. Then from (i) of Definition 5.4, we have $\deg(M_1) \gtrsim \deg(M_2)$.

Case 3. $M_1 \equiv \mathcal{E}_n[\cdots\mathcal{E}_i[\cdots\mathcal{E}_1[\mu\alpha.M]\cdots]\cdots] \rhd_{st} \mathcal{E}_n[\cdots\mathcal{E}'_i[\cdots\mathcal{E}_1[\mu\alpha.M]\cdots]\cdots] \equiv M_2$, where $\mathcal{E}_i \rhd_{st} \mathcal{E}'_i$ $(1 \leq i \leq n)$:

Case 3-1. $\mathcal{E}_i \equiv \mathcal{E}_1 \equiv \mathcal{E}'[\mu\beta.M_3]$ and M contains a subterm of the form $[\alpha]V$: from the induction hypothesis, we have either $\deg(\mathcal{E}_1) \succ \deg(\mathcal{E}'_1)$ or $\deg(\mathcal{E}_1) \gtrsim \deg(\mathcal{E}'_1)$. Hence, we have $\deg(M_1) \succ \deg(M_2)$ or $\deg(M_1) \gtrsim \deg(M_2)$ by (III) of Definition 5.1 or (iii) of Definition 5.4, respectively.

Case 3-2. Otherwise, $\deg(M_1)$ has no change from Definition 5.2. Then we have $\deg(M_1) \gtrsim \deg(M_2)$ by (I) of Definition 5.1. □

The following lemma essentially means that $\rhd_{st}$ is strongly normalizable even for untyped terms, from which Proposition 3.13 can be verified.

Lemma 5.7. *If we have* $M_1 \rhd_{st} M_2$, *then* $\mathrm{Deg}(M_1) \gg \mathrm{Deg}(M_2)$.

Proof. By induction on the length of M_1 and following the similar case analysis to Lemma 5.6, we prove that $\mathrm{Deg}(M_1) \gg \mathrm{Deg}(M_2)$ if $M_1 \rhd_{st} M_2$. From Definition 5.3, we only show the case where M_1 is in the form of $\mathcal{E}_n[\mathcal{E}_{n-1}[\cdots\mathcal{E}_1[\mu\alpha.M]\cdots]]$. Other cases can be confirmed by the induction hypothesis together with (VI) of Definition 5.5.

Case 1. $M_1 \equiv \mathcal{E}_n[\cdots\mathcal{E}_1[\mu\alpha.M]\cdots] \rhd_{st} \mathcal{E}_n[\cdots\mathcal{E}_2[\mu\alpha.M[\mathcal{E}_1/\alpha]]\cdots] \equiv M_2$: since $\deg(M_1) \succ \deg(M_2)$, we have $\mathrm{Deg}(M_1) \gg \mathrm{Deg}(M_2)$ by (IV) of Definition 5.5.

Case 2. $M_1 \equiv \mathcal{E}_n[\cdots\mathcal{E}_1[\mu\alpha.M]\cdots] \rhd_{st} \mathcal{E}_n[\cdots\mathcal{E}_1[\mu\alpha.M']\cdots] \equiv M_2$, where $M \rhd_{st} M'$: we have the two cases where successive redexes P_k in M is reduced or not.

Case 2-1. Successive redexes P_k in M is reduced: if $M_1 \equiv P_{m+1} \equiv \mathcal{E}^{m+1}[\mu\alpha_{m+1}.N_{m+1}] \rhd_{st} \mathcal{E}^{m+1}[\mu\alpha_{m+1}.N'_{m+1}] \equiv M_2$, where N'_{m+1} is obtained from N_{m+1} by reducing a subterm $P_{k\leq m}$ of N_{m+1} in the same way as Case 1 above, then $\deg(M_1) \succ \deg(M_2)$. Hence, $\mathrm{Deg}(M_1) \gg \mathrm{Deg}(M_2)$ by (IV) of Definition 5.5.

Case 2-2. Non-successive redexes $\mathcal{E}'[\mu\beta.M_3]$ is reduced in M, where $\mathcal{E}'[\mu\beta.M_3] \rhd_{st} \mu\beta.M_3[\mathcal{E}'/\beta]$:

Case 2-2-1. $\mathcal{E}'$ of $\mathcal{E}'[\mu\beta.M_3]$ in M contains successive redexes P_k:

Case 2-2-1-1. $\beta \notin FN(M_3)$: we have $\mathcal{E}'[\mu\beta.M_3] \rhd_{st} \mu\beta.M_3$, and then $\deg(M_1) \succ \deg(M_2)$. Hence, $\mathrm{Deg}(M_1) \gg \mathrm{Deg}(M_2)$ from (IV) and (VI) of Definition 5.5.

Case 2-2-1-2. $\beta \in FN(M_3)$: after the reduction $\mathcal{E}'[\mu\beta.M_3] \rhd_{st} \mu\beta.M_3[\mathcal{E}'/\beta]$, the successive redexes P_k in $\mathcal{E}'$ is copied as the number of the free occurrences of β in M_3 so that $\deg(M_1) \gtrsim \deg(M_2)$.

Let M contain l free occurrences of $[\alpha]$. An interesting case is that $\mathcal{E}'$ contains $l_1 \leq l$ free occurrences of $[\alpha]$ and M_3 contains $l_2 \geq 1$ free occurrences of $[\beta]$. In this case, we have $(l_1 \times l_2)$ free occurrences of $[\alpha]$ in $M_3[\mathcal{E}'/\beta]$, and then M' contains $(l + l_1 \times (l_2 - 1))$ free occurrences of $[\alpha]$, which is greater than l when $l_2 > 1$. This means that in the root tree of the whole tree, *i.e.*, $\deg(M_1)$, the branching number at some node of $\deg(M_1)$ changes from l to $(l + l_1 \times (l_2 - 1))$, which is caused by $(l_1 \times (l_2 - 1))$ times copying the corresponding subtree P_k in M_1. However, we have $\mathrm{Deg}(M) \gg \mathrm{Deg}(M')$ from the induction hypothesis. Therefore, we have $\mathrm{Deg}(M_1) \gg \mathrm{Deg}(M_2)$ by (V) of Definition 5.5.

Case 2-2-2. $\mathcal{E}'$ of $\mathcal{E}'[\mu\beta.M_3]$ in M contains no successive redexes: even after the reduction $\mathcal{E}'[\mu\beta.M_3] \rhd_{st} \mu\beta.M_3[\mathcal{E}'/\beta]$, $\deg(M_1)$ has no change, since $\mathcal{E}'$ contains no P_k. From the induction hypothesis, we have $\mathrm{Deg}(M) \gg \mathrm{Deg}(M')$, and hence $\mathrm{Deg}(M_1) \gg \mathrm{Deg}(M_2)$ by (VI) of Definition 5.5.

Case 3. $M_1 \equiv \mathcal{E}_n[\cdots\mathcal{E}_i[\cdots\mathcal{E}_1[\mu\alpha.M]\cdots]\cdots] \rhd_{st} \mathcal{E}_n[\cdots\mathcal{E}'_i[\cdots\mathcal{E}_1[\mu\alpha.M]\cdots]\cdots] \equiv M_2$, where $\mathcal{E}_i \rhd_{st} \mathcal{E}'_i$ $(1 \leq i \leq n)$:

Case 3-1. $\mathcal{E}_i \equiv \mathcal{E}_1 \equiv \mathcal{E}'[\mu\beta.M_3]$ and M contains a subterm of the form $[\alpha]V$: from Lemma 5.6, we have either $\deg(\mathcal{E}_1) \succ \deg(\mathcal{E}'_1)$ or $\deg(\mathcal{E}_1) \gtrsim \deg(\mathcal{E}'_1)$.

In the first subcase of $\deg(\mathcal{E}_1) \succ \deg(\mathcal{E}'_1)$, we have $\deg(M_1) \succ \deg(M_2)$, and hence $\mathrm{Deg}(M_1) \gg \mathrm{Deg}(M_2)$ from (IV) of Definition 5.5.

In the second subcase of $\deg(\mathcal{E}_1) \gtrsim \deg(\mathcal{E}'_1)$, we have $\mathrm{Deg}(\mathcal{E}_1) \gg \mathrm{Deg}(\mathcal{E}'_1)$ from the induction hypothesis. If $\deg(\mathcal{E}_1) = \deg(\mathcal{E}'_1)$, then $\mathrm{Deg}(M_1) \gg \mathrm{Deg}(M_2)$ by (VI) of Definition 5.5. Otherwise, $\deg(\mathcal{E}'_1)$ has a copied subtree. Then we have $\mathrm{Deg}(M_1) \gg \mathrm{Deg}(M_2)$ from $\mathrm{Deg}(\mathcal{E}_1) \gg \mathrm{Deg}(\mathcal{E}'_1)$ and (V) of Definition 5.5.

Case 3-2. Otherwise, $\deg(M_1)$ has no change from Definition 5.2. From the induction hypothesis, we have $\mathrm{Deg}(\mathcal{E}_i) \gg \mathrm{Deg}(\mathcal{E}_i')$, and then $\mathrm{Deg}(M_1) \gg \mathrm{Deg}(M_2)$ by (VI) of Definition 5.5.

□

Acknowledgements. I am grateful to Susumu Hayashi, Yukiyoshi Kameyama, and the members of the Proof Animation Group for helpful discussions. Nakazawa Koji and Tatuta Makoto commented on the paper in the previous version. I would like to thank Gilles Barthe and Morten Heine B. Sørensen for pointing out their paper in the revised version [8]. The author is also grateful to anonymous referees for their constructive comments and suggestions.

REFERENCES

[1] Y. Andou, A Normalization-Procedure for the First Order Classical Natural Deduction with Full Logical Symbols, *Tsukuba J. Math.* **19** (1995) 153-162.

[2] Y. Andou, *Church–Rosser property of a simple reduction for full first order classical natural deduction* (submitted).

[3] Y. Andou, *Strong normalization of classical natural deduction*, Logic Colloquium 2000. *Bull. Symbolic Logic* (to appear).

[4] F. Barbanera and S. Berardi, *Extracting Constructive Context from Classical Logic via Control-like Reductions.* Springer, *Lecture Notes in Comput. Sci.* **664** (1993) 45-59.

[5] H.P. Barendregt, *The Lambda Calculus, Its Syntax and Semantics*, Revised edition. North–Holland (1984).

[6] H.P. Barendregt, *Lambda Calculi with Types.* Oxford University Press, *Handbook of Logic in Comput. Sci.* **2** (1992) 117-309.

[7] G. Barthe and M.H. Sørensen, *Domain-Free Pure Type Systems.* Springer, *Lecture Notes in Comput. Sci.* **1234** (1997) 9-20.

[8] G. Barthe and M.H. Sørensen, *Domain-Free Pure Type Systems*, the revised version of [7].

[9] K. Baba, S. Hirokawa and K. Fujita, Parallel Reduction in Type-Free $\lambda\mu$-Calculus, in *The 7th Asian Logic Conference* (1999).

[10] K. Baba, S. Hirokawa and K. Fujita, *Parallel Reduction in Type-Free $\lambda\mu$-Calculus.* Elsevier, Amsterdam, *Electron. Notes Theoret. Comput. Sci.* **42** (2001) 52-66.

[11] L. Damas and R. Milner, Principal type-schemes for functional programs, in *Proc. the 9th Annual ACM Symposium on Principles of Programming Languages* (1982) 207-212.

[12] R. Davies and F. Pfenning, A Modal Analysis of Staged Computation, in *Proc. the 23rd Annual ACM Symposium of Principles of Programming Languages* (1996).

[13] R. Davies and F. Pfenning, *A Modal Analysis of Staged Computation*, Technical Report CMU-CS-99-153 (1999).

[14] M. Felleisen, D.P. Friedman, E. Kohlbecker and B. Duba, Reasoning with Continuations, in *Proc. the Annual IEEE Symposium on Logic in Computer Science* (1986) 131-141.

[15] P.C. Fischer, Turing Machines with Restricted Memory Access. *Inform. and Control* **9** (1966) 364-379.

[16] K. Fujita, *On embedding of Classical Substractural Logics*, RIMS 918. Research Institute for Mathematical Sciences, Kyoto University (1995) 178-195.

[17] K. Fujita, *Calculus of Classical Proofs I.* Springer, *Lecture Notes in Comput. Sci.* **1345** (1997) 321-335.

[18] K. Fujita, *Polymorphic Call-by-Value Calculus based on Classical Proofs.* Springer, *Lecture Notes in Artificial Intelligence* **1476** (1998) 170-182.

[19] K. Fujita, *Explicitly Typed $\lambda\mu$-Calculus for Polymorphism and Call-by-Value.* Springer, *Lecture Notes in Comput. Sci.* **1581** (1999) 162-176.
[20] K. Fujita, *Type Inference for Domain-Free* $\lambda 2$, Technical Report in Computer Science and Systems Engineering CSSE–5. Kyushu Institute of Technology (1999).
[21] K. Fujita and A. Schubert, *Partially Typed Terms between Church-Style and Curry-Style.* Springer, *Lecture Notes in Comput. Sci.* **1872** (2000) 505-520.
[22] T.G. Griffin, A Formulae-as-Types Notion of Control, in *Proc. the 17th Annual ACM Symposium on Principles of Programming Languages* (1990) 47-58.
[23] Ph. de Groote, *A CPS-Translation for the $\lambda\mu$-Calculus.* Springer, *Lecture Notes in Comput. Sci.* **787** (1994) 85-99.
[24] Ph. de Groote, *A Simple Calculus of Exception Handling.* Springer, *Lecture Notes in Comput. Sci.* **902** (1995) 201-215.
[25] R. Harper, B.F. Duba and D. MacQueen, Typing First-Class Continuations in ML. *J. Funct. Programming* **3** (1993) 465-484.
[26] R. Harper and M. Lillibridge, ML with callcc is unsound. *The Types Form*, 8 July (1991).
[27] R. Harper and M. Lillibridge, Explicit polymorphism and CPS conversion, in *Proc. the 20th Annual ACM Symposium on Principles of Programming Languages* (1993) 206-219.
[28] R. Harper and M. Lillibridge, Polymorphic type assignment and CPS conversion. *LISP and Symbolic Computation* **6** (1993) 361-380.
[29] R. Harper and J.C. Mitchell, On The Type Structure of Standard ML. *ACM Trans. Programming Languages and Systems* **15** (1993) 210-252.
[30] M. Hofmann, Sound and complete axiomatisations of call-by-value control operators. *Math. Structures Comput. Sci.* **5** (1995) 461-482.
[31] M. Hofmann and T. Streicher, Continuation models are universal for $\lambda\mu$-calculus, in *Proc. the 12th Annual IEEE Symposium on Logic in Computer Science* (1997) 387-395.
[32] J.E. Hopcroft and J.D. Ullman, *Introduction to Automata Theory, Languages and Computation.* Addison–Wesley (1979).
[33] W. Howard, The Formulae-as-Types Notion of Constructions, *To H.B. Curry: Essays on combinatory logic, lambda-calculus, and formalism.* Academic Press (1980) 479-490.
[34] A.J. Kfoury, J. Tiuryn and P. Urzyczyn, An Analysis of ML Typability. *J. Assoc. Comput. Mach.* **41** (1994) 368-398.
[35] S. Kobayashi, Monads as modality. *Theoret. Comput. Sci.* **175** (1997) 29-74.
[36] X. Leroy, Polymorphism by name for references and continuations, in *Proc. the 20th Annual ACM Symposium of Principles of Programming Languages* (1993) 220-231.
[37] S. Martini and A. Massini, A Computational Interpretation of Modal Proofs, *Proof Theory of Modal Logic.* Kluwer (1996) 213-241.
[38] A. Meyer and M. Wand, Continuation Semantics in Typed Lambda-Calculi. Springer, *Lecture Notes in Comput. Sci.* **193** (1985) 219-224.
[39] J.C. Mitchell, Polymorphic Type Inference and Containment. *Inform. and Comput.* **76** (1988) 211-249.
[40] R. Milner, A Theory of Type Polymorphism in Programming. *J. Comput. System Sci.* **17** (1978) 348-375.
[41] M.L. Minsky, Recursive Unsolvability of Post's Problem of "TAG" and Other Topics in Theory of Turing Machines. *Ann. of Math.* **74** (1961) 437-455.
[42] E. Moggi, Notions of computation and monads. *Inform. and Comput.* **93** (1991) 55-92.
[43] C.R. Murthy, An Evaluation Semantics for Classical Proofs, in *Proc. the 6th Annual IEEE Symposium on Logic in Computer Science* (1991) 96-107.
[44] C.-H.L. Ong, A Semantic View of Classical Proofs: Type-Theoretic, Categorical, and Denotational Characterizations, *Linear Logic '96 Tokyo Meeting* (1996).
[45] C.-H.L. Ong and C.A. Stewart, A Curry-Howard Foundation for Functional Computation with Control, in *Proc. the 24th Annual ACM Symposium of Principles of Programming Languages* (1997).
[46] M. Parigot, *$\lambda\mu$-Calculus: An Algorithmic Interpretation of Classical Natural Deduction.* Springer, *Lecture Notes in Comput. Sci.* **624** (1992) 190-201.

[47] M. Parigot, *Classical Proofs as Programs.* Springer, *Lecture Notes in Comput. Sci.* **713** (1993) 263-276.
[48] M. Parigot, Proofs of Strong Normalization for Second Order Classical Natural Deduction. *J. Symbolic Logic* **62** (1997) 1461-1479.
[49] G. Plotkin, Call-by-Name, Call-by-Value and the λ-Calculus. *Theoret. Comput. Sci.* **1** (1975) 125-159.
[50] D. Prawitz, *Natural Deduction: A Proof-Theoretical Study.* Almqvist & Wiksell (1965).
[51] D. Prawitz, Ideas and Results in Proof Theory, in *Proc. the 2nd Scandinavian Logic Symposium*, edited by N.E. Fenstad. North–Holland (1971) 235-307.
[52] N.J. Rehof and M.H. Sørensen, *The λ_Δ-Calculus.* Springer, *Lecture Notes in Comput. Sci.* **789** (1994) 516-542.
[53] A. Schubert, *Second-order unification and type inference for Church-style*, Tech. Report TR 97-02 (239). Institute of Informatics, Warsaw University (1997).
[54] A. Schubert, Second-order unification and type inference for Church-style, in *Proc. the ACM Symposium on Principles of Programming Languages* (1998) 279-288.
[55] T. Streicher and B. Reus, Continuation semantics: Abstract machines and control operators. *J. Funct. Programming* (to appear).
[56] M. Takahashi, Parallel Reductions in λ-Calculus. *J. Symbolic Comput.* **7** (1989) 113-123.
[57] J. Tiuryn, *Type Inference Problems: A Survey.* Springer, *Lecture Notes in Comput. Sci.* **452** (1990) 105-120.
[58] J.B. Wells, Typability and Type Checking in the Second-Order λ-Calculus Are Equivalent and Undecidable, in *Proc. IEEE Symposium on Logic in Computer Science* (1994) 176-185.

Communicated by Z. Ésik.
Received September 8, 1999. Accepted March 6, 2001.