数理解析研究所講究録 1437 巻 2005 年 163-173

A sound and complete CPS-translation for $\lambda\mu$-calculus
– Extended abstract –

Ken-etsu Fujita (藤田 憲悦)
Gunma University (群馬大学)
fujita@cs.gunma-u.ac.jp

Abstract

We provide a bijective CPS-translation for type-free $\lambda\mu$-calculus. This method can be naturally carried over to second order typed $\lambda\mu$-calculus, which leads to a bijective CPS-translation between classical proofs and intuitionistic proofs. We also investigate an abstract machine for $\lambda\mu$-calculus, which handles explicitly environments.

1 Introduction

The term CPS-translation, in general, denotes a program translation method into continuation passing style that is the meaning of the program as a function taking the rest of the computation. The method has been studied for program transformation, definitional interpreter, or denotational semantics [Reyn93].

On the other hand, according to Griffin [Grif90], a CPS-translation corresponds to a logical embedding from classical logic into intuitionistic logic under the Formulae-as-Types correspondence [How80]. Parigot [Pari92, Pari93, Pari97] introduced the $\lambda\mu$-calculus from the viewpoint of classical logic, and established an extension of the Curry-Howard isomorphism [Grif90, Murt91]. A semantics of monomorphic $\lambda\mu$-calculus has been investigated recently from the viewpoint of continuations. There have been noteworthy investigations including Hofmann-Streicher [HS97], Streicher-Reus [SR98], and Selinger [Seli01]: In terms of a category of continuations, a continuation semantics of simply typed $\lambda\mu$-calculus is proved to be sound and complete for any $\lambda\mu$-theory [HS97]. Under the control category, it is established that an isomorphism between call-by-name and call-by-value $\lambda\mu$-calculi with conjunction and disjunction types [Seli01]. The category of negated domains is introduced as a model of type free $\lambda\mu$-calculus [SR98]. Streicher-Reus also remarked that a CPS-translation naïvely based on Plotkin [Plot75] cannot validate (η)-rule. All of the work involve a novel CPS-translation which requires, at least, products as a primitive notion, so that the extensionality, (η)-rule can be validated by the surjective pairing, as observed in [Fuji03a].

An analysis on the calculi without type restrictions reveals core properties of the CPS-translation and the universe consisting of the image of the translation. Continuations are handled as a list of denotations, and formalized as a pair consisting of a denotation and a continuation in this order. The study on the type free cases also makes clear the distinction between λ-calculus and $\lambda\mu$-calculus, from the viewpoint of continuations: an λ-abstraction is viewed as a function taking only the first component of such a pair, and on the other hands, an μ-abstraction is interpreted as an λ-abstraction over continuations. This paper is a revised version of both work [Fuji03b] presented at the 6th International Conference on Typed Lambda Calculi and Applications, TLCA 2003, Valencia, Spain, June 2003; and at the 5th Symposium on Algebra and Computation, Tokyo Metropolitan University, October 2003. The method in this article can be naturally carried over to second order typed $\lambda\mu$-calculus, which leads to a bijective CPS-translation between classical proofs and intuitionistic proofs.

2 CPS-Translation of type free λ-calculus with extensionality

We first study already known CPS-translations [Plot75, HS97] and yet another translation with `let`-expressions [Fuji05]. This section also serves as a gentle introduction to CPS-translations.

2.1 Plotkin's call-by-name CPS-translation and (η)-rule

The definitions of terms and reduction rules are given to the extensional λ-calculi, respectively, denoted by Λ, $\Lambda^{\langle\rangle}$ and $\Lambda^{\texttt{let}}$.

Definition 1 **(λ-calculus Λ)**

$$\Lambda \ni M \quad ::= \quad x \mid \lambda x.M \mid MM$$

(β) $(\lambda x.M_1)M_2 \to M_1[x := M_2]$

(η) $\lambda x.Mx \to M$ *if* $x \notin FV(M)$

Definition 2 **(λ-calculus with surjective pairing $\Lambda^{\langle\rangle}$)**

$$\Lambda^{\langle\rangle} \ni M \quad ::= \quad x \mid \lambda x.M \mid MM \mid \langle M, M\rangle \mid \pi_1(M) \mid \pi_2(M)$$

(β) $(\lambda x.M_1)M_2 \to M_1[x := M_2]$

(η) $\lambda x.Mx \to M$ *if* $x \notin FV(M)$

(π) $\pi_i\langle M_1, M_2\rangle \to M_i \quad (i = 1, 2)$

(sp) $\langle \pi_1(M), \pi_2(M)\rangle \to M$

Definition 3 **(λ-calculus with `let` $\Lambda^{\texttt{let}}$)**

$$\Lambda^{\texttt{let}} \ni M \quad ::= \quad x \mid \lambda x.M \mid MM \mid \langle M, M\rangle \mid \texttt{let}\ \langle x, x\rangle = M\ \texttt{in}\ M$$

(β) $(\lambda x.M_1)M_2 \to M_1[x := M_2]$

(η) $\lambda x.Mx \to M$ *if* $x \notin FV(M)$

(let) $\texttt{let}\ \langle x_1, x_2\rangle = \langle M_1, M_2\rangle\ \texttt{in}\ M \to M[x_1 := M_1, x_2 := M_2]$

($\texttt{let}_\eta$) $\texttt{let}\ \langle x_1, x_2\rangle = M_1\ \texttt{in}\ M[x := \langle x_1, x_2\rangle] \to M[x := M_1]$ *if* $x_1, x_2 \notin FV(M)$

The term $M_1[x := M_2]$ denotes the result of substituting M_2 for the free occurrences of x in M_1. $FV(M)$ stands for the set of free variables in M. The term $M[x_1 := M_1, x_2 := M_2]$ denotes the result of substituting simultaneously M_1 and M_2 respectively for the free occurrences of x_1 and x_2 in M. The one step reduction relation is denoted by $\to_R$ where R consists of (β), (η), etc. We write $\to_R^+$ and $\to_R^*$ to denote the transitive closure and the reflexive and transitive closure of $\to_R$, respectively. We employ the notation $=_R$ to indicate the symmetric, reflexive and transitive closure of $\to_R$. The binary relation $\equiv$ denotes the syntactic identity under renaming of bound variables.

A term M is always evaluated in a certain context $\mathcal{E}[\ ]$ roughly understood as a term with a hole or the rest of the computation. Such a context can be formalized as a function $\lambda x.\mathcal{E}[x]$ and called a continuation with respect to M. Then an application of a continuation to an argument means filling the argument with the hole of the evaluation context. A CPS-translation of a term M gives a function $\underline{M}$ such that the function explicitly takes, as an argument, the continuation with respect to M.

Definition 4 (Plotkin's call-by-name CPS-translation [Plot75])

(i) $\underline{x} = x$

(ii) $\underline{M_1 M_2} = \lambda k.\underline{M_1}(\lambda m.m\underline{M_2}k)$

(iii) $\underline{\lambda x.M} = \lambda k.k(\lambda x.\underline{M})$

According to Plotkin's definition, the second clause says that a continuation of the function M_1 is informally a context in the form of $[\ ]\underline{M_2}k$ where k is a continuation of M_1M_2. Here this context is formalized as the code of the pair consisting of $\underline{M_2}$ and k in this order. That is, a continuation k is to be understood as the form of $\langle \pi_1 k, \pi_2 k\rangle$. Then we can grasp an informal meaning $[\![M_1M_2]\!] \sim \lambda k.[\![M_1]\!]\langle [\![M_2]\!], k\rangle$. The third clause means filling $\lambda x.\underline{M}$ with the hole of the evaluation context of $\lambda x.M$, which is in the form of a pair from the second clause. Then the hole of $[\ ](\pi_1 k)(\pi_2 k)$ is filled by $\lambda x.\underline{M}$. Hence the third clause can be understood as $[\![\lambda x.M]\!] \sim \lambda k.(\lambda x.[\![M]\!])(\pi_1 k)(\pi_2 k)$. An λ-abstraction is interpreted as a function taking, as an argument, a first component of such a pair. One may find less distinction between _ and $[\![\ ]\!]$. However, considering an interpretation of the (η)-rule reveals a deep gap between the two definitions. Let $x \notin FV(M)$.

$[\![\lambda x.Mx]\!] \sim \lambda k.(\lambda x.\lambda k.[\![M]\!]\langle \lambda k.xk, k\rangle)(\pi_1 k)(\pi_2 k) \to_{\beta\eta}^+ \lambda k.[\![M]\!]\langle \pi_1 k, \pi_2 k\rangle$

The above computation including (β) and (η) means that we cannot interpret (η)-rule following the original definition of Plotkin, since adding the surjective pairing to the (β) and (η) calculus breaks down the Church-Rosser property as proved by Klop [Bare84]. In other words, we should prepare a target calculus with the surjective pairing in order to validate

(η)-rule [HS97, Seli01] along the line of Plotkin's idea. This method also discussed in the previous version [Fuji03b] interprets m-input Curried function as un-Curried function with $(m+1)$-component, under β-reductions, as follows:

$[\![\lambda x_1 \ldots x_m.xM_1 \ldots M_n]\!] \sim \lambda k.x\langle[\![M_1]\!], \ldots, \langle[\![M_n]\!], k\rangle \ldots\rangle\theta$

where θ is a substitution $[x_1 := \pi_1 k, x_2 := \pi_1(\pi_2 k), \ldots, x_m := \pi_1(\pi_2^{m-1}k)]$. Here, the first m components contain the denotations of m arguments, respectively, and the last component is for the rest continuation. Although this method of course works well as done in [HS97, Seli01], we introduce here yet another way such that projections are packed into an let-expression, as follows:

Definition 5 (CPS-translation : $\Lambda \rightarrow \Lambda^{\mathtt{let}}$) (i) $[\![x]\!] = x$

(ii) $[\![\lambda x.M]\!] = \lambda a.(\mathtt{let}\ \langle x, b\rangle = a\ \mathtt{in}\ [\![M]\!]b)$

(iii) $[\![M_1 M_2]\!] = \lambda a.[\![M_1]\!]\langle[\![M_2]\!], a\rangle$

This modification seems to be trivial where the let-expression is not syntactic sugar. However, the use of let-expressions makes it possible to handle the substitution information in a suspended way, in general, environments elegantly, and to simplify extremely technical matters on the completeness[1], comparing with the previous version [Fuji03b].

3 Type free $\lambda\mu$-calculus

Secondly we study type free $\lambda\mu$-calculus from the view point of the CPS-translation introduced in the previous section.

3.1 Extensional $\lambda\mu$-calculus and CPS-translation

We give the definition of type free $\lambda\mu$-calculus with (η). The syntax of the $\lambda\mu$-terms is defined from variables, λ-abstraction, application, or μ-abstraction over names denoted by α, where a term in the form of $[\alpha]M$ is called a named term.

Definition 6 ($\lambda\mu$-calculus $\Lambda\mu$)

$$\Lambda\mu \ni M ::= x \mid \lambda x.M \mid MM \mid \mu\alpha.N \qquad N ::= [\alpha]M$$

(β) $(\lambda x.M_1)M_2 \rightarrow M_1[x := M_2]$

(η) $\lambda x.Mx \rightarrow M$ *if* $x \notin FV(M)$

(μ) $(\mu\alpha.N)M \rightarrow \mu\alpha.N[\alpha \Leftarrow M]$

(μ_β) $\mu\alpha.[\beta](\mu\gamma.N) \rightarrow \mu\alpha.N[\gamma := \beta]$

(μ_η) $\mu\alpha.[\alpha]M \rightarrow M$ *if* $\alpha \notin FN(M)$

[1]This point is also important in the polymorphic case.

$FN(M)$ stands for the set of free names in M. The $\lambda\mu$-term $N[\alpha \Leftarrow M]$ denotes a term obtained by replacing each subterm of the form $[\alpha]M'$ in N with $[\alpha](M'M)$. This operation is inductively defined as follows:

1. $x[\alpha \Leftarrow M] = x$
2. $(\lambda x.M_1)[\alpha \Leftarrow M] = \lambda x.M_1[\alpha \Leftarrow M]$
3. $(M_1M_2)[\alpha \Leftarrow M] = (M_1[\alpha \Leftarrow M])(M_2[\alpha \Leftarrow M])$
4. $(\mu\beta.N)[\alpha \Leftarrow M] = \mu\gamma.N[\beta := \gamma][\alpha \Leftarrow M]$ where γ is a fresh name
5. $([\beta]M_1)[\alpha \Leftarrow M] = \begin{cases} [\beta]((M_1[\alpha \Leftarrow M])M), & \text{for } \alpha \equiv \beta \\ [\beta](M_1[\alpha \Leftarrow M]), & \text{otherwise} \end{cases}$

The term $M[\alpha \Leftarrow M_1, \ldots, M_n]$ denotes $M[\alpha \Leftarrow M_1]\cdots[\alpha \Leftarrow M_n]$.

The binary relation $=_{\lambda\mu}$ over $\Lambda\mu$ denotes the symmetric, reflexive and transitive closure of the one step reduction relation, i.e., the equivalence relation induced from the reduction rules.

Definition 7 (CPS-Translation : $\Lambda\mu \to \Lambda^{\mathtt{let}}$) (i) $[\![x]\!] = x$

(ii) $[\![\lambda x.M]\!] = \lambda a.(\mathtt{let}\ \langle x, b\rangle = a\ \mathtt{in}\ [\![M]\!]b)$

(iii) $[\![M_1M_2]\!] = \lambda a.[\![M_1]\!]\langle [\![M_2]\!], a\rangle$

(iv) $[\![\mu a.[b]M]\!] = \lambda a.[\![M]\!]b$

Proposition 1 (Soundness) *Let $M_1, M_2 \in \Lambda\mu$.*
If we have $M_1 =_{\lambda\mu} M_2$ then $[\![M_2]\!] =_{\lambda^{\mathtt{let}}} [\![M_2]\!]$.

Proof. By induction on the derivation of $M_1 =_{\lambda\mu} M_2$, together with the following facts:

$$\begin{array}{rcl} [\![M_1]\!][x := [\![M_2]\!]] & = & [\![M_1[x := M_2]]\!] \\ [\![M_1[\alpha \Leftarrow M_2]]\!] & \to_\beta^* & [\![M_1]\!][\alpha := \langle [\![M_2]\!], \alpha\rangle] \end{array}$$

1. Case of (β):
   $[\![(\lambda x.M_1)M_2]\!] = \lambda a.((\lambda b.\mathtt{let}\ \langle x, c\rangle = b\ \mathtt{in}\ [\![M_1]\!]c)\langle [\![M_2]\!], a\rangle)$
   $\to_\beta \lambda a.(\mathtt{let}\ \langle x, c\rangle = \langle [\![M_2]\!], a\rangle\ \mathtt{in}\ [\![M_1]\!]c)$
   $\to_{\mathtt{let}} \lambda a.[\![M_2]\!]a[x := [\![M_1]\!]] = \lambda a.[\![M_1[x := M_2]]\!]a$
   $\to_\eta [\![M_1[x := M_2]]\!]$
2. Case of (η):
   $[\![\lambda x.Mx]\!] = \lambda a.(\mathtt{let}\ \langle x, b\rangle = a\ \mathtt{in}\ (\lambda c.[\![M]\!]\langle x, c\rangle)b)$
   $\to_\beta \lambda a.(\mathtt{let}\ \langle x, b\rangle = a\ \mathtt{in}\ [\![M]\!]\langle x, b\rangle)$
   $\to_{\mathtt{let}_\eta} \lambda a.[\![M]\!]a$
   $\to_\eta [\![M]\!]$

3. Case of (μ):
$[\![(\mu\alpha.[\beta]M_1)M_2]\!] = \lambda a.(\lambda\alpha.[\![M_1]\!]\beta)\langle[\![M_2]\!], a\rangle$
$\rightarrow_\beta \lambda a.([\![M_1]\!]\beta)[\alpha := \langle[\![M_2]\!], a\rangle]$
$$=_\beta \begin{cases} \lambda a.[\![M_1[\alpha \Leftarrow M_2]]\!]\beta[\alpha := a] = [\![\mu\alpha.[\beta](M_1[\alpha \Leftarrow M_2])]\!] & \text{if } \alpha \not\equiv \beta \\ \lambda a.[\![M_1[\alpha \Leftarrow M_2]]\!]\langle[\![M_2]\!], \alpha\rangle[\alpha := a] = [\![\mu\alpha.[\beta]((M_1[\alpha \Leftarrow M_2])M_2)]\!] & \text{if } \alpha \equiv \beta \end{cases}$$

4. Case of (μ_η):
$[\![\mu\alpha.[\alpha]M]\!] = \lambda\alpha.[\![M]\!]\alpha$
$\rightarrow_\eta [\![M]\!]$

5. Case of (μ_β):
$[\![\mu\alpha.[\beta](\mu\gamma.[\delta]M)]\!] = \lambda\alpha.(\lambda\gamma.[\![M]\!]\delta)\beta$
$\rightarrow_\beta \lambda\alpha.[\![M]\!]\delta[\gamma := \beta] = [\![\mu\alpha.[\delta]M[\gamma := \beta]]\!]$ □

3.2 Inverse translation and completeness

We define a set of $\Lambda^{\mathtt{let}}$-terms called *Univ*, which is the image of the CPS-translation closed under reductions.

$$Univ \overset{\text{def}}{=} \{P \in \Lambda^{\mathtt{let}} \mid [\![M]\!] \rightarrow^*_{\lambda^{\mathtt{let}}} P \text{ for some } M \in \Lambda\mu\}$$

We introduce a grammar $\mathcal{R}$ that describes *Univ*. Let $n \geq 0$. Then we write $\langle M_0, M_1, \ldots, M_n\rangle$ for $\langle M_0, \langle M_1, \ldots, M_n\rangle\rangle$, and $\langle M\rangle \equiv M$.

$$\begin{array}{rcl} \mathcal{R} & ::= & x \mid \lambda a.\mathcal{R}\langle\mathcal{R}_1, \ldots, \mathcal{R}_n, a\rangle \\ & & \mid \lambda a.(\mathtt{let}\ \langle x, a\rangle = \langle\mathcal{R}_1, \ldots, \mathcal{R}_n, a\rangle\ \mathtt{in}\ \mathcal{R}\langle\mathcal{R}_1, \ldots, \mathcal{R}_n, a\rangle) \end{array}$$

Lemma 1 (1) *The category $\mathcal{R}$ is closed under reduction rules of $\lambda^{\mathtt{let}}$.*

(2) *Univ* $\subseteq \mathcal{R}$

Proof. (1) Let $R, R_i \in \mathcal{R}$. Then we have the facts that $R[x := R_1] \in \mathcal{R}$ and $R[a := \langle R_1, \ldots, R_n, b\rangle] \in \mathcal{R}$.
(2) $[\![M]\!] \in \mathcal{R}$ and $\mathcal{R}$ is closed under reduction rules. □

We introduce an inverse translation $\natural$ from $\mathcal{R}$ back to $\Lambda\mu$.

Definition 8 (Inverse Translation $\natural : \mathcal{R} \rightarrow \Lambda\mu$)

(i) $x^\natural = x$

(ii) $(\lambda a.R\langle R_1, \ldots, R_n, b\rangle)^\natural = \mu a.[b](R^\natural R_1^\natural \ldots R_n^\natural)$

(iii) $(\lambda a.(\mathtt{let}\ \langle x, b\rangle = \langle R_1, \ldots, R_m, c\rangle\ \mathtt{in}\ S\langle S_1, \ldots, S_n, d\rangle))^\natural$
$= \mu a.[c]((\lambda x.(\lambda b.S\langle S_1, \ldots, S_n, d\rangle)^\natural)R_1^\natural \ldots R_m^\natural)$

Lemma 2 (1) *Let $M \in \Lambda\mu$. Then we have that $[\![M]\!]^\natural \to^*_{\mu_\eta} M$.*

(2) *Let $P \in \mathcal{R}$. Then we have $[\![P^\natural]\!] \to^*_\beta P$.*

Proof. By induction on the structure of $M \in \Lambda\mu$ and $R \in \mathcal{R} \supseteq$ *Univ*, respectively.

(1) (i) $[\![\lambda x.M]\!]^\natural = \{\lambda a.(\mathtt{let}\ \langle x,b\rangle = a\ \mathtt{in}\ [\![M]\!]b)\}^\natural$
$= \mu a.[a]\lambda x.\{\lambda b.[\![M]\!]b\}^\natural = \mu a.[a]\lambda x.\mu b.[b][\![M]\!]^\natural$
$\to^+_{\mu_\eta} \lambda x.M$

(ii) $[\![M_1 M_2]\!]^\natural = \{\lambda a.[\![M_1]\!]\langle[\![M_2]\!],a\rangle\}^\natural$
$= \mu a.[a][\![M_1]\!]^\sharp[\![M_2]\!]^\natural \to_{\mu_\eta} M_1 M_2$

(iii) $[\![\mu a.[b]M]\!]^\natural = \{\lambda a.[\![M]\!]b\}^\natural$
$= \mu a.[b][\![M]\!]^\natural \to^*_{\mu_\eta} \mu a.[b]M$ by the induction hypothesis.

(2) (ii)

$[\![(\lambda a.R\langle R_1,\dots,R_n,b\rangle)^\natural]\!] = [\![\mu a.[b](R^\natural R_1^\natural \dots R_n^\natural)]\!]$
$\to^+_\beta \lambda a.(\lambda a'.[\![R^\natural]\!]\langle[\![R_1^\natural]\!],\dots,[\![R_n^\natural]\!],a'\rangle)b$
$\to_\beta \lambda a.[\![R^\natural]\!]\langle[\![R_1^\natural]\!],\dots,[\![R_n^\natural]\!],b\rangle$
$\to^*_\beta \lambda a.R\langle R_1,\dots,R_n,b\rangle$ by the induction hypotheses.

(iii)

$[\![(\lambda a.(\mathtt{let}\ \langle x,b\rangle = \langle R_1,\dots,R_m,c\rangle\ \mathtt{in}\ S\langle S_1,\dots,S_n,d\rangle))^\natural]\!]$
$= [\![\mu a.[c]((\lambda x.(\lambda b.S\langle S_1,\dots,S_n,d\rangle)^\natural)R_1^\natural\dots R_m^\natural)]\!]$
$= \lambda a.[\![(\lambda x.(\lambda b.S\langle S_1,\dots,S_n,d\rangle)^\natural)R_1^\natural\dots R_m^\natural]\!]c$
$\to^+_\beta \lambda a.(\lambda e.[\![\lambda x.(\lambda b.S\langle S_1,\dots,S_n,d\rangle)^\natural]\!]\langle[\![R_1^\natural]\!],\dots,[\![R_m^\natural]\!],e\rangle)c$
$\to_\beta \lambda a.[\![\lambda x.(\lambda b.S\langle S_1,\dots,S_n,d\rangle)^\natural]\!]\langle[\![R_1^\natural]\!],\dots,[\![R_m^\natural]\!],c\rangle$
$= \lambda a.(\lambda e.\mathtt{let}\ \langle x,f\rangle = e\ \mathtt{in}\ [\![(\lambda b.S\langle S_1,\dots,S_n,d\rangle)^\natural]\!]f)\langle[\![R_1^\natural]\!],\dots,[\![R_m^\natural]\!],c\rangle$
$\to_\beta \lambda a.(\mathtt{let}\ \langle x,f\rangle = \langle[\![R_1^\natural]\!],\dots,[\![R_m^\natural]\!],c\rangle\ \mathtt{in}\ [\![(\lambda b.S\langle S_1,\dots,S_n,d\rangle)^\natural]\!]f)$
$\to^*_\beta \lambda a.(\mathtt{let}\ \langle x,f\rangle = \langle R_1,\dots,R_m,c\rangle\ \mathtt{in}\ (\lambda b.S\langle S_1,\dots,S_n,d\rangle)f)$
by the induction hypotheses
$\to_\beta \lambda a.(\mathtt{let}\ \langle x,b\rangle = \langle R_1,\dots,R_m,c\rangle\ \mathtt{in}\ S\langle S_1,\dots,S_n,d\rangle)$ □

Lemma 3 *Let $R, R_1,\dots,R_n \in \mathcal{R}$.*
Then we have $(R[a := \langle R_1,\dots,R_n,a\rangle])^\natural = R^\natural[a \Leftarrow R_1^\natural,\dots,R_n^\natural]$

Proof. By induction on the structure of $\mathcal{R}$. □

Proposition 2 (Completeness) *Let $P, Q \in \mathcal{R}$.*

(1) *If $P \to_\beta Q$ then $P^\natural \to^+_{\mu\mu_\beta} Q^\natural$.*

(2) *If* $P \to_{\eta} Q$ *then* $P^{\natural} \to_{\mu_\eta} Q^{\natural}$.

(3) *If* $P \to_{\mathtt{let}} Q$ *then* $P^{\natural} \to^{+}_{\beta\mu\mu_\beta} Q^{\natural}$.

(4) *If* $P \to_{\mathtt{let}_\eta} Q$ *then* $P^{\natural} =_{\beta\eta\mu\mu_\eta} Q^{\natural}$.

Proof.

(1) Let K be $\langle S_1, \ldots, S_n, d\rangle$.

$$(\lambda a.(\lambda b.R\langle R_1, \ldots, R_m, c\rangle)K)^{\natural} = \mu a.[d]((\mu b.[c]R^{\natural}R_1^{\natural} \ldots R_m^{\natural})S_1^{\natural} \ldots S_n^{\natural})$$
$$\to^{*}_{\mu} \mu a.[d](\mu b.[c]R^{\natural}R_1^{\natural} \ldots R_m^{\natural}[b \Leftarrow S_1^{\natural}, \ldots, S_n^{\natural}])$$
$$\to_{\mu_\beta} \mu a.[c]R^{\natural}R_1^{\natural} \ldots R_m^{\natural}[b \Leftarrow S_1^{\natural}, \ldots, S_n^{\natural}][b := d]$$
$$= (\lambda a.R\langle R_1, \ldots, R_m, c\rangle[b := \langle S_1, \ldots, S_n, d\rangle])^{\natural}$$

(2) $(\lambda a.Ra)^{\natural} = \mu a.[a]R^{\natural} \to_{\mu_\eta} R^{\natural}$

(3) $(\lambda a.(\mathtt{let}\ \langle x, b\rangle = \langle R_0, R_1, \ldots, R_m, c\rangle\ \mathtt{in}\ S\langle S_1, \ldots, S_n, d\rangle))^{\natural}$

$$= \mu a.[c]((\lambda x.(\mu b.[d]S^{\natural}S_1^{\natural} \ldots S_n^{\natural}))R_0^{\natural}R_1^{\natural} \ldots R_m^{\natural})$$
$$\to_{\beta} \mu a.[c]((\mu b.[d]S^{\natural}S_1^{\natural} \ldots S_n^{\natural})[x := R_0^{\natural}]R_1^{\natural} \ldots R_m^{\natural})$$
$$\to^{*}_{\mu} \mu a.[c](\mu b.[d]S^{\natural}S_1^{\natural} \ldots S_n^{\natural}[x := R_0^{\natural}][b \Leftarrow R_1^{\natural}, \ldots, R_m^{\natural}])$$
$$\to_{\mu_\beta} \mu a.[d]S^{\natural}S_1^{\natural} \ldots S_n^{\natural}[x := R_0^{\natural}][b \Leftarrow R_1^{\natural}, \ldots, R_m^{\natural}][b := c]$$
$$= (\lambda a.S\langle S_1, \ldots, S_n, d\rangle[x := R_0, b := \langle R_1, \ldots, R_m, c\rangle])^{\natural}$$

(4) $(\mathtt{let})$ can play the role of $(\mathtt{let}_\eta)$ except for the following case:
$\lambda a.\mathtt{let}\ \langle x, b\rangle = c\ \mathtt{in}\ R\langle R_1, \ldots, R_m, d\rangle[e := \langle x, b\rangle] \to \lambda a.R\langle R_1, \ldots, R_m, d\rangle[e := c]$, where $x, b \notin FV(RR_1 \ldots R_m d)$.
We also have $\mu\alpha.M =_{\lambda\mu} \lambda x.(\mu\alpha.M)x =_{\lambda\mu} \lambda x.\mu\alpha.M[\alpha \Leftarrow x]$.
Then we have as follows:

$$(\lambda a.\mathtt{let}\ \langle x, b\rangle = c\ \mathtt{in}\ R\langle R_1, \ldots, R_m, d\rangle[e := \langle x, b\rangle])^{\natural}$$
$$= \mu a.[c](\lambda x.\mu b.([d](R^{\natural}R_1^{\natural} \ldots R_m^{\natural}))[e \Leftarrow x][e := b])$$
$$=_{\lambda\mu} \mu a.[c]\mu b.([d](R^{\natural}R_1^{\natural} \ldots R_m^{\natural}))[e := b]$$
$$\to_{\mu_\beta} \mu a.[d]R^{\natural}R_1^{\natural} \ldots R_m^{\natural}[e := b][b := c]$$
$$= \mu a.[d]R^{\natural}R_1^{\natural} \ldots R_m^{\natural}[e := c] = (\lambda a.R\langle R_1, \ldots, R_m, d\rangle[e := c])^{\natural}$$

□

Theorem 1 (i) *Let* $M_1, M_2 \in \Lambda\mu$. $M_1 =_{\lambda\mu} M_2$ *if and only if* $[\![M_1]\!] =_{\lambda^{\mathtt{let}}} [\![M_2]\!]$.

(ii) *Let* $P_1, P_2 \in \mathcal{R}$. $P_1 =_{\lambda^{\mathtt{let}}} P_2$ *if and only if* $P_1^{\natural} =_{\lambda\mu} P_2^{\natural}$.

Proof.

(i) From Propositions 1 and 2 and Lemma 2 (1).

(ii) From Propositions 1 and 2 and Lemma 2 (2). □

Corollary 1 *Univ* $= \mathcal{R}$

Proof. We have $Univ \subseteq \mathcal{R}$ from Lemma 1. Let $P \in \mathcal{R}$. Then $[\![P^\natural]\!] \to_\beta^* P$ from Lemma 2, and hence we have $P \in Univ$. □

Corollary 2 *The inverse translation* $\natural : Univ \to \Lambda\mu$ *is bijective, in the following sense:*

(1) *If we have* $P_1^\natural =_{\lambda\mu} P_2^\natural$ *then* $P_1 =_{\lambda^{\mathrm{let}}} P_2$ *for* $P_1, P_2 \in Univ$.

(2) *For any* $M \in \Lambda\mu$, *we have some* $P \in Univ$ *such that* $P^\natural =_{\lambda\mu} M$.

4 Abstract machine with explicit environment

Finally we briefly introduce an abstract machine for $\lambda\mu$-calculus, which handles environments explicitly and is motivated by our target calculus with `let`-expressions.

There exists a well-known connection between continuation passing style [Seli98, SR98] and abstract machines [Plot75, Bier98, deGr98]. For instance, according to [SR98], we have relations between denotation and closure; coninuation and stack; and environments.

Continuation Denotational Semantics	denotation D $[\![\ ]\!] : \Lambda \times E \to D$ $D = [K \to R]$	continuation K $D \times K$	environment E $Var \to D$ $Cvar \to K$
Abstract Machine	closure $Clos$ $\Lambda \times E$	stack S $Clos \times S$	environment E $Var \to Clos$

where Λ is a set of terms, and R is a domain of responses.

Due to [SR98], let $D = R^K$ be the solution of $K = R^K \times K$ where R is non-empty. Let Env be a set of environments, such that $Env = (\mathtt{Var} \to D) \times (\mathtt{Name} \to K)$. The semantic function $[\![\ ]\!]_D : \Lambda\mu \times Env \to D$ is defined as follows [SR98]:

1. $[\![x]\!]_D\ e\ k = e(x)\ k$
2. $[\![\lambda x.M]\!]_D\ e\ \langle d, k\rangle = [\![M]\!]_D\ (e[x := d])\ k$
3. $[\![M_1 M_2]\!]_D\ e\ k = [\![M_1]\!]_D\ e\ \langle [\![M_2]\!]_D\ e, k\rangle$
4. $[\![\mu\alpha.[\beta]M]\!]_D\ e\ k = [\![M]\!]_D\ (e[\alpha := k])\ (e[\alpha := k](\beta))$

We introduce here an abstract machine with a modification, such that the environment explicitly handles substitution information consisting of continuations. The machine has configurations of the form $\langle [M, E], K\rangle$, where $[M, E]$ is the closure consisting of a term M (instruction) and the environment E, and K is the continuation. Environments are defined by continuations (a list of substitution information where :: denotes `cons`), and continuations consist of a closure and a continuation.

Environment (list of continuations)

$$E ::= \mathtt{nil} \mid ((\langle x, k\rangle = K) :: E \mid (k = K) :: E$$

Continuation (list of closures)

$$K ::= k \mid \langle cl, K\rangle \mid E(k) \mid \mathtt{snd}(K)$$

Closure

$cl ::= [M, E] \mid E(x) \mid \texttt{fst}(K)$

The transition function $\Rightarrow$ specifies how to execute the terms, in the sense that one step execution transforms the configuration $\langle [M, E], K \rangle$.

1. $\langle [x, E], K \rangle \Rightarrow \langle E(x), K \rangle$
2. $\langle [\lambda x.M, E], K \rangle \Rightarrow \langle [M, E_1], \texttt{snd}(K) \rangle$
   where $E_1 = ((\langle x, k \rangle = K) :: E)$ with fresh variable k
3. $\langle [M_1 M_2, E], K \rangle \Rightarrow \langle [M_1, E], \langle [M_2, E], K \rangle \rangle$
4. $\langle [\mu\alpha.[\beta]M, E], K \rangle \Rightarrow \langle [M, E_1], E_1(\beta) \rangle$
   where $E_1 = ((\alpha = K) :: E)$

Moreover, environments are also handled by the transition function $\Rightarrow_e$.

(i) $((k = K) :: E)(x') \Rightarrow_e E(x')$

(ii) $$((\langle x, k \rangle = K) :: E)(x') \Rightarrow_e \begin{cases} \texttt{fst}(K) & \text{if } x \equiv x' \text{ and } K \text{ is a pair} \\ \texttt{fst}(E(k_1)) & \text{if } x \equiv x' \text{ and } K \text{ is a variable } k_1 \\ E(x') & \text{otherwise} \end{cases}$$

(iii) $$((k = K) :: E)(k') \Rightarrow_e \begin{cases} E(k_1) & \text{if } k \equiv k' \text{ and } K \text{ is a variable } k_1 \\ K & \text{if } k \equiv k' \text{ and } K \text{ is a pair} \\ E(k') & \text{otherwise} \end{cases}$$

(iv) $$((\langle x, k \rangle = K) :: E)(k') \Rightarrow_e \begin{cases} \texttt{snd}(E(k_1)) & \text{if } k \equiv k' \text{ and } K \text{ is a variable } k_1 \\ \texttt{snd}(K) & \text{if } k \equiv k' \text{ and } K \text{ is a pair} \\ E(k') & \text{otherwise} \end{cases}$$

where $\texttt{fst}\langle cl, K \rangle \Rightarrow_e cl$, $\texttt{snd}\langle cl, K \rangle \Rightarrow_e K$, and $\langle \texttt{fst}(K), \texttt{snd}(K) \rangle \Rightarrow_e K$.

References

[Bare84] H. P. Barendregt: *The Lambda Calculus, Its Syntax and Semantics* (revised edition), North-Holland, 1984.

[Bier98] G. M. Bierman: A computational interpretation of the $\lambda\mu$-calculus, Lecture Notes in Computer Science 1450, pp. 336-345, 1998.

[deGr98] Ph. de Groote: An environment machine for the $\lambda\mu$-calculus, *Math. Struct. in Compu. Science*, 1998.

[Fuji03a] K. Fujita: Simple Models of Type Free $\lambda\mu$-Calculus, *Computer Software*, Japan Society for Software Science and Technology, Vol. 20, No. 3, pp. 73–79, 2003.

[Fuji03b] K. Fujita: A sound and complete CSP-translation for $\lambda\mu$-Calculus, Lecture Notes in Computer Science 2701, pp. 120–134, 2003.

[Fuji05] K. Fujita: Galois embedding from polymorphic types into existential types, Lecture Notes in Computer Science 3461, pp. 194–208, 2005.

[Grif90] T. G. Griffin: A Formulae-as-Types Notion of Control, *Proc. the 17th Annual ACM Symposium on Principles of Programming Languages*, pp. 47–58, 1990.

[HS97] M. Hofmann and T. Streicher: Continuation models are universal for $\lambda\mu$-calculus, *Proc. the 12th Annual IEEE Symposium on Logic in Computer Science*, pp. 387–395, 1997.

[How80] W. Howard: The Formulae-as-Types Notion of Constructions, in: *To H.B.Curry: Essays on combinatory logic, lambda-calculus, and formalism*, Academic Press, pp. 479–490, 1980.

[Murt91] C. R. Murthy: An Evaluation Semantics for Classical Proofs, *Proc. the 6th Annual IEEE Symposium on Logic in Computer Science*, pp. 96–107, 1991.

[Pari92] M. Parigot: $\lambda\mu$-Calculus: An Algorithmic Interpretation of Classical Natural Deduction, Lecture Notes in Computer Science 624, pp. 190–201, 1992.

[Pari93] M. Parigot: Classical Proofs as Programs, Lecture Notes in Computer Science 713, pp. 263–276, 1993.

[Pari97] M. Parigot: Proofs of Strong Normalization for Second Order Classical Natural Deduction, *J. Symbolic Logic*, Vol. 62, No. 4, pp. 1461–1479, 1997.

[Plot75] G. Plotkin: Call-by-Name, Call-by-Value and the λ-Calculus, *Theoretical Computer Science*, Vol. 1, pp. 125–159, 1975.

[Reyn93] J. C. Reynolds: The discoveries of continuation, *Lisp and Symbolic Computation*, Vol. 6, pp. 233–247, 1993.

[Seli98] P. Selinger: An implementation of the call-by-name $\lambda\mu\nu$-calculus, manuscript, 1998.

[Seli01] P. Selinger: Control Categories and Duality: on the Categorical Semantics of the Lambda-Mu Calculus, *Math. Struct. in Compu. Science*, Vol. 11, pp. 207–260, 2001.

[SR98] T. Streicher and B. Reus: Classical Logic, Continuation Semantics and Abstract Machines, *J. Functional Programming*, Vol. 8, No. 6, pp. 543–572, 1998.