

λ-Calculus with Lazy Lists

– Extended Abstract –

Ken-etsu Fujita (藤田 憲悦)

Shimane University (島根大学)

Department of Mathematics and Computer Science

Matsue 690-8504, Japan

fujiken@cis.shimane-u.ac.jp

Abstract

Into λ-calculus we introduce lazy lists $\vec{a}$ whose naïve meaning is an infinite list consisting of variables, $(a_0, a_1, a_2, \dots)$. It is shown that there exist maps which form a Galois connection from Parigot's $\lambda\mu$ -calculus to the λ-calculus with lazy list. The translations form not only an equational correspondence but also a reduction correspondence between the two calculi.

1 Introduction

We introduce lazy lists into λ-calculus. The introduction of infinite lists is motivated by a study on denotational semantics of type-free $\lambda\mu$ -calculus [Pari92, Pari97, BHF99, BHF01].

Given domains $U \times U \cong U \cong [U \rightarrow U]$ such as in Lambek-Scott [LS86], we have established a continuation denotational semantics of type-free $\lambda\mu$ -calculus [Fuji02], which formally coincides with the CPS-translation [HS97, SR98, Fuji01] followed by the direct denotational semantics of the λ-calculus [Scot72, Stoy77]. See also the literature [HS97, SR98, Seli01] for continuation semantics of $\lambda\mu$ -calculus.

This article shows that there exists a one-to-one correspondence between the $\lambda\mu$ -calculus and the λ-calculus with lazy lists.

2 $\vec{\lambda}$ -calculus

We have two kinds of variables, the traditional variables in the λ-calculus denoted by x and variables for lazy lists denoted by $\vec{a}$. Our intended meaning is that $\vec{a}$ denotes an infinite list of variables, $\vec{a} \simeq (a_0, a_1, a_2, \dots)$. The denotational meaning of $\vec{a}$ would be given by elements of domain E^ω which is a solution of the domain equation $D \cong D \times D$. From this intension, the expression $M\vec{a}$ says that M is a function which can accept infinite inputs, and $\lambda\vec{a}.M$ is a function characterized by $D^D \cong D^{D \times D} \cong D^{D^D}$, that is, $\lambda\vec{a}.\dots M\vec{a}\dots$ can behave like $\lambda x.\lambda\vec{a}.\dots Mx\vec{a}\dots$. Under this informal meaning, potentially infinite applications of β -reduction should be performed as follows:

$$\begin{aligned}
& (\lambda \vec{a}. \dots (M_1 \vec{a}) \dots) M \\
& \simeq (\lambda a_0 a_1 \dots \dots (M_1 a_0 a_1 \dots) \dots) M \\
& =_{\beta} \lambda a_1 a_2 \dots \dots (M_1 M a_1 a_2 \dots) \dots \\
& \simeq \lambda \vec{a}. \dots (M_1 M \vec{a}) \dots
\end{aligned}$$

Following this intended meaning, we define the λ -calculus with infinite lists as follows. A term in the form of $\vec{a}$ is called a lazy list.

Definition 1 ($\vec{\lambda}$ -calculus)

$$\vec{\Lambda} \ni M ::= x \mid \vec{a} \mid \lambda x. M \mid \lambda \vec{a}. M \mid M M$$

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

$$(\eta) \lambda x. M x = M \text{ if } x \notin FV(M)$$

$$(\vec{\beta}) (\lambda \vec{a}. M_1) M_2 = \begin{cases} M_1[\vec{a} := M_2] & \text{if } M_2 \text{ is in the form of a lazy list} \\ \lambda \vec{a}. M_1[\vec{a} := M_2] & \text{otherwise} \end{cases}$$

$$(\vec{\eta}) \lambda \vec{a}. M \vec{a} = M \text{ if } \vec{a} \notin FV(M)$$

The term $M_1[x := M_2]$ denotes the usual capture-free substitution of M_2 for x in M_1 . The term $M_1[\vec{a} := M_2]$ indicates the capture-free substitution defined in the following:

$$(i) x[\vec{a} := M] = x$$

$$(ii) \vec{b}[\vec{a} := M] = \begin{cases} M & \text{if } \vec{b} \equiv \vec{a} \text{ and } M \text{ is a lazy list} \\ M \vec{b} & \text{if } \vec{b} \equiv \vec{a} \text{ and } M \text{ is not a lazy list} \\ \vec{b} & \text{otherwise} \end{cases}$$

$$(iii) (\lambda x. M_1)[\vec{a} := M] = \lambda x. M_1[\vec{a} := M]$$

$$(iv) (\lambda \vec{b}. M_1)[\vec{a} := M] = \lambda \vec{b}. M_1[\vec{a} := M]$$

$$(v) (M_1 M_2)[\vec{a} := M] = \begin{cases} ((M_1[\vec{a} := M]) M) M_2 & \text{if } M_2 \equiv \vec{a} \text{ and } M \text{ is not a lazy list} \\ (M_1[\vec{a} := M])(M_2[\vec{a} := M]) & \text{otherwise} \end{cases}$$

The axiom $(\vec{\beta})$ says that a function which can accept an infinite list has taken an infinite list in the case where M_2 is in the form of a lazy list. In the case where M_2 is not in the form of a lazy list, $(\vec{\beta})$ means that a function which can accept an infinite list has taken only a finite input, so that we still have $\lambda \vec{a}$ even after this. $(\vec{\eta})$ says that $\lambda \vec{a}. M \vec{a}$ is an infinite η -expansion of M .

We write $\vec{\Lambda} \vdash M_1 = M_2$ or $(\beta, \eta, \vec{\beta}, \vec{\eta}) \vdash M_1 = M_2$ if $M_1 = M_2$ is derived from the axioms (β) , (η) , $(\vec{\beta})$, or $(\vec{\eta})$. As an abbreviation, we may write $M_1 =_{\vec{\Lambda}} M_2$ for this. We adopt a rewriting theory of $\vec{\Lambda}$ by rewriting the left-hand side of each axiom to the corresponding right-hand side. The binary relation $\rightarrow$, $\rightarrow^+$, or $\rightarrow^*$ denotes the one-step rewriting, the transitive closure of $\rightarrow$, or the reflexive and transitive closure of $\rightarrow$, respectively.

Proposition 1 (1) $\vec{\Lambda} \vdash \lambda x. x = \lambda \vec{a}. \vec{a}$

$$(2) \quad \vec{\Lambda} \vdash \lambda x. \lambda \vec{a}. M[\vec{a} := x] = \lambda \vec{a}. M$$

Proof. (1) $\lambda \vec{a}. \vec{a}$ can be regarded as an infinite η -expansions of $\lambda x. x$:

$$\lambda \vec{a}. \vec{a} =_{\eta} \lambda x. (\lambda \vec{a}. \vec{a})x =_{\beta} \lambda x. (\lambda \vec{a}. x\vec{a}) =_{\eta} \lambda x. x$$

(2) The abstraction by λx can be absorbed in the infinite λ -abstraction by $\lambda \vec{a}$:

Let $x \notin FV(M)$.

$$\lambda \vec{a}. M =_{\eta} \lambda x. (\lambda \vec{a}. M)x =_{\beta} \lambda x. \lambda \vec{a}. M[\vec{a} := x]$$

□

3 Relationship between $\lambda\mu$ -calculus and $\vec{\lambda}$ -calculus

We show that the $\vec{\lambda}$ -calculus is a conservative extension over Parigot's $\lambda\mu$ -calculus [Pari92, Pari97].

Definition 2 ($\lambda\mu$ -calculus)

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

$$(\beta) \quad (\lambda x. M_1)M_2 = M_1[x := M_2]$$

$$(\eta) \quad \lambda x. Mx = M \text{ if } x \notin FV(M)$$

$$(\mu) \quad (\mu\alpha. M_1)M_2 = \mu\alpha. M_1[\alpha \Leftarrow M_2]$$

$$(\mu\beta) \quad [\alpha](\mu\beta. M) = M[\beta := \alpha]$$

$$(\mu_{\eta}) \quad \mu\alpha. [\alpha]M = M \text{ if } \alpha \notin FV(M)$$

The $\lambda\mu$ -term $M_1[\alpha \Leftarrow M_2]$ denotes a term obtained by replacing each subterm of the form $[\alpha]M$ in M_1 with $[\alpha](MM_2)$. This operation is inductively defined as follows:

$$1. \quad x[\alpha \Leftarrow M] = x$$

$$2. \quad (\lambda x. M_1)[\alpha \Leftarrow M] = \lambda x. M_1[\alpha \Leftarrow M]$$

$$3. \quad (M_1M_2)[\alpha \Leftarrow M] = (M_1[\alpha \Leftarrow M])(M_2[\alpha \Leftarrow M])$$

$$4. \quad (\mu\beta. N)[\alpha \Leftarrow M] = \mu\beta. N[\alpha \Leftarrow M]$$

$$5. \quad ([\beta]M_1)[\alpha \Leftarrow M] = \begin{cases} [\beta]((M_1[\alpha \Leftarrow M])M), & \text{for } \alpha \equiv \beta \\ [\beta](M_1[\alpha \Leftarrow M]), & \text{otherwise} \end{cases}$$

Definition 3 *Translation* $\lceil \cdot \rceil : \Lambda\mu \rightarrow \vec{\Lambda}$

$$1. \quad \lceil x \rceil = x$$

$$2. \quad \lceil \lambda x. M \rceil = \lambda x. \lceil M \rceil$$

$$3. \quad \lceil M_1M_2 \rceil = \lceil M_1 \rceil \lceil M_2 \rceil$$

$$4. \lceil \mu\alpha.M \rceil = \lambda\vec{\alpha}.\lceil M \rceil$$

$$5. \lceil [\alpha]M \rceil = \lceil M \rceil \vec{\alpha}$$

Lemma 1 Let $M, N \in \Lambda\mu$.

$$\lceil M[\alpha \leftarrow N] \rceil = \lceil M \rceil [\vec{\alpha} := \lceil N \rceil]$$

Proof. By induction on the structure of M . Noted that $\lceil N \rceil$ cannot be a lazy list. $\square$

Proposition 2 If $M_1 \rightarrow_{\Lambda\mu} M_2$, then $\lceil M_1 \rceil \rightarrow_{\vec{\Lambda}} \lceil M_2 \rceil$.

Proof. By induction on the derivation of $M_1 \rightarrow_{\Lambda\mu} M_2$. We show some of the base cases.

Case of (μ) :

$$\begin{aligned} \lceil (\mu\alpha.M)N \rceil &= (\lambda\vec{\alpha}.\lceil M \rceil)[\lceil N \rceil] \\ &\rightarrow_{\beta} \lambda\vec{\alpha}.\lceil M \rceil[\vec{\alpha} := \lceil N \rceil] \text{ since } \lceil N \rceil \text{ is not a lazy list} \\ &= \lambda\vec{\alpha}.\lceil M[\alpha \leftarrow N] \rceil = \lceil \mu\alpha.M[\alpha \leftarrow N] \rceil \end{aligned}$$

Case of (β) :

$$\begin{aligned} \lceil (\lambda x.M)N \rceil &= (\lambda x.\lceil M \rceil)[\lceil N \rceil] \\ &\rightarrow_{\beta} \lceil M \rceil[x := \lceil N \rceil] = \lceil M[x := N] \rceil \end{aligned}$$

$\square$

Definition 4 Translation $\lfloor \cdot \rfloor : \vec{\Lambda} \rightarrow \Lambda\mu$

$$(i) \lfloor x \rfloor = x$$

$$(ii) \lfloor \vec{a} \rfloor = [a](\lambda x.x)$$

$$(iii) \lfloor \lambda x.M \rfloor = \lambda x.\lfloor M \rfloor$$

$$(iv) \lfloor \lambda \vec{a}.M \rfloor = \mu a.\lfloor M \rfloor$$

$$(v) \lfloor M_1 M_2 \rfloor = \begin{cases} [a]\lfloor M_1 \rfloor & \text{if } M_2 \equiv \vec{a} \text{ for some } \vec{a} \\ \lfloor M_1 \rfloor \lfloor M_2 \rfloor & \text{otherwise} \end{cases}$$

Lemma 2 (i) Let $M \in \vec{\Lambda}$.

$$\lfloor M \rfloor [a := b] = \lfloor M[\vec{a} := \vec{b}] \rfloor$$

(ii) Let $M, N \in \vec{\Lambda}$ where N is not a lazy list.

$$\lfloor M \rfloor [a \leftarrow \lfloor N \rfloor] \rightarrow_{\beta}^* \lfloor M[\vec{a} := N] \rfloor$$

Proof. By induction on the structure of M . We show some cases for (ii).

Case of $M \equiv \vec{a}$:

$$\begin{aligned} \lfloor \vec{a} \rfloor [a \leftarrow \lfloor N \rfloor] &= ([a](\lambda x.x))[a \leftarrow \lfloor N \rfloor] = [a](\lambda x.x)\lfloor N \rfloor \\ &\rightarrow_{\beta} [a]\lfloor N \rfloor = \lfloor N \vec{a} \rfloor = \lfloor \vec{a}[\vec{a} := N] \rfloor \text{ since } N \text{ is not a lazy list.} \end{aligned}$$

Case of $M \equiv M_1 M_2$:

We show the subcase M_2 of $\vec{a}$ here.

$$\lfloor M_1 M_2 \rfloor [a \leftarrow \lfloor N \rfloor] = \lfloor M_1 \vec{a} \rfloor [a \leftarrow \lfloor N \rfloor]$$

$$\begin{aligned}
&= ([a][M_1])[a \Leftarrow [N]] \\
&= [a]([M_1][a \Leftarrow [N]])[N] \\
&\rightarrow_{\beta}^* [a]([M_1[\vec{a} := N]])[N] \quad \text{by the induction hypothesis} \\
&= [a]([M_1[\vec{a} := N]N]) \\
&= [(M_1[\vec{a} := N]N)\vec{a}] = [(M_1\vec{a})[\vec{a} := N]]
\end{aligned}$$

□

Proposition 3 *Let $M_1, M_2 \in \vec{\Lambda}$.*

If $M_1 \rightarrow_{\vec{\Lambda}} M_2$ then $[M_1] \rightarrow_{\Lambda\mu}^+ [M_2]$.

Proof. By induction on the derivation of $M_1 \rightarrow_{\vec{\Lambda}} M_2$.

Case of $(\vec{\beta})$ where N is not a lazy list:

$$\begin{aligned}
&[(\lambda\vec{a}.M)N] = (\mu a.[M])[N] \\
&\rightarrow_{\mu} \mu a.[M][a \Leftarrow [N]] \\
&\rightarrow_{\beta}^* \mu a.[M[\vec{a} := N]] \quad \text{by Lemma 2} \\
&= [\lambda\vec{a}.M[\vec{a} := N]]
\end{aligned}$$

Case of $(\vec{\beta})$ where N is a lazy list:

$$\begin{aligned}
&[(\lambda\vec{a}.M)\vec{b}] = [b](\mu a.[M]) \\
&\rightarrow_{\mu\beta} [M][a := b] = [M[\vec{a} := \vec{b}]]
\end{aligned}$$

Case of (β) :

$$\begin{aligned}
&[(\lambda x.M)N] = (\lambda x.[M])[N] \\
&\rightarrow_{\beta} [M][x := [N]] = [M[x := N]] \quad \text{by Lemma 2}
\end{aligned}$$

□

Proposition 4 *The maps $\lceil \cdot \rceil : \Lambda\mu \rightarrow \vec{\Lambda}$ and $\lfloor \cdot \rfloor : \vec{\Lambda} \rightarrow \Lambda\mu$ establish a one-to-one correspondence between $\Lambda\mu$ and $\vec{\Lambda}$:*

(i) *For any $M \in \Lambda\mu$, $M = \lfloor \lceil M \rceil \rfloor$.*

(ii) *For any $M \in \vec{\Lambda}$, $\lceil \lfloor M \rfloor \rceil \rightarrow_{\beta}^* M$.*

Proof. By induction on the structure of M . For (ii) we show one of the base cases.

Case of $M \equiv \vec{a}$:

$$\lceil [\vec{a}] \rceil = \lceil [a](\lambda x.x) \rceil = (\lambda x.x)\vec{a} \rightarrow_{\beta} \vec{a}$$

□

Definition 5 (Sabry-Walder [SW96])

The maps $\lceil \cdot \rceil$ and $\lfloor \cdot \rfloor$ form a Galois connection from $\Lambda\mu$ to $\vec{\Lambda}$ whenever $M \rightarrow_{\Lambda\mu}^ [P]$ if and only if $\lceil M \rceil \rightarrow_{\vec{\Lambda}}^* P$.*

$$\begin{array}{ccc}
M \in \Lambda\mu & \xrightarrow{\lceil \cdot \rceil} & \vec{\Lambda} \ni \lceil M \rceil \\
\downarrow \text{\scriptsize $*\downarrow_{\Lambda\mu}$} & & \downarrow \text{\scriptsize $*\downarrow_{\vec{\Lambda}}$} \\
[P] \in \Lambda\mu & \xleftarrow{\lfloor \cdot \rfloor} & \vec{\Lambda} \ni P
\end{array}$$

It can be confirmed that the maps $\lceil \cdot \rceil : \Lambda\mu \rightarrow \vec{\Lambda}$ and $\lfloor \cdot \rfloor : \vec{\Lambda} \rightarrow \Lambda\mu$ form a Galois connection by Propositions 2, 3, 4, and the following proposition:

Proposition 5 (Sabry-Wadler [SW96])

The maps $\lceil \cdot \rceil$ and $\lfloor \cdot \rfloor$ form a Galois connection from $\Lambda\mu$ to $\vec{\Lambda}$ if and only if conditions hold:

- (i) $M \rightarrow_{\Lambda\mu}^* \lfloor \lceil M \rceil \rfloor$,
- (ii) $\lceil \lfloor P \rfloor \rceil \rightarrow_{\vec{\Lambda}}^* P$,
- (iii) $M_1 \rightarrow_{\Lambda\mu}^* M_2$ implies $\lceil M_1 \rceil \rightarrow_{\vec{\Lambda}}^* \lceil M_2 \rceil$, and
- (iv) $P_1 \rightarrow_{\vec{\Lambda}}^* P_2$ implies $\lfloor P_1 \rfloor \rightarrow_{\Lambda\mu}^* \lfloor P_2 \rfloor$.

See also the following diagrams:

$$\begin{array}{ll}
 \text{(i)} & \begin{array}{ccc} M \in \Lambda\mu & \xrightarrow{\lceil \cdot \rceil} & \vec{\Lambda} \ni \lceil M \rceil \\ \downarrow *_{\Lambda\mu} & & \parallel \\ \lfloor \lceil M \rceil \rfloor \in \Lambda\mu & \xleftarrow{\lfloor \cdot \rfloor} & \vec{\Lambda} \ni \lfloor \lceil M \rceil \rfloor \end{array} \\
 \text{(ii)} & \begin{array}{ccc} \lceil P \rceil \in \Lambda\mu & \xrightarrow{\lceil \cdot \rceil} & \vec{\Lambda} \ni \lceil \lfloor P \rfloor \rceil \\ \parallel & & \downarrow *_{\vec{\Lambda}} \\ \lfloor P \rfloor \in \Lambda\mu & \xleftarrow{\lfloor \cdot \rfloor} & \vec{\Lambda} \ni P \end{array} \\
 \text{(iii)} & \begin{array}{ccc} M_1 \in \Lambda\mu & \xrightarrow{\lceil \cdot \rceil} & \vec{\Lambda} \ni \lceil M_1 \rceil \\ \downarrow *_{\Lambda\mu} & & \downarrow *_{\vec{\Lambda}} \\ M_2 \in \Lambda\mu & \xrightarrow{\lceil \cdot \rceil} & \vec{\Lambda} \ni \lceil M_2 \rceil \end{array} \\
 \text{(iv)} & \begin{array}{ccc} \lfloor P_1 \rfloor \in \Lambda\mu & \xleftarrow{\lfloor \cdot \rfloor} & \vec{\Lambda} \ni P_1 \\ \downarrow *_{\Lambda\mu} & & \downarrow *_{\vec{\Lambda}} \\ \lfloor P_2 \rfloor \in \Lambda\mu & \xleftarrow{\lfloor \cdot \rfloor} & \vec{\Lambda} \ni P_2 \end{array}
 \end{array}$$

Proposition 6 (i) (Conservative extension) *Let $M_1, M_2 \in \Lambda\mu$.*

If we have $\lceil M_1 \rceil =_{\vec{\Lambda}} \lceil M_2 \rceil$, then $M_1 =_{\Lambda\mu} M_2$.

(ii) (Galois connection)

The maps $\lceil \cdot \rceil : \Lambda\mu \rightarrow \vec{\Lambda}$ and $\lfloor \cdot \rfloor : \vec{\Lambda} \rightarrow \Lambda\mu$ form a Galois connection from $\lambda\mu$ -calculus to the $\vec{\lambda}$ -calculus.

(iii) (Equational/reduction correspondence)

(1) $M_1 =_{\Lambda\mu} M_2$ if and only if $\lceil M_1 \rceil =_{\bar{\Lambda}} \lceil M_2 \rceil$.

In particular, $M_1 \rightarrow_{\Lambda\mu}^+ M_2$ if and only if $\lceil M_1 \rceil \rightarrow_{\bar{\Lambda}}^+ \lceil M_2 \rceil$.

(2) $P_1 =_{\bar{\Lambda}} P_2$ if and only if $\lfloor P_1 \rfloor =_{\Lambda\mu} \lfloor P_2 \rfloor$.

Proof. (i) From Proposition 4.

(ii) From Propositions 2, 3, 4, and 5.

(iii) From (ii). □

Acknowledgement

Thanks to Izumi Takeuti for helpful comments on this work.

References

- [BHF99] K. Baba, S. Hirokawa, and K. Fujita: Parallel Reduction in Type-Free $\lambda\mu$ -Calculus, *The 7th Asian Logic Conference*, 1999.
- [BHF01] K. Baba, S. Hirokawa, and K. Fujita: Parallel Reduction in Type-Free $\lambda\mu$ -Calculus, *Electronic Notes in Theoretical Computer Science*, Vol. 42, pp. 52–66, 2001.
- [Fuji01] K. Fujita: Simple Model of Type Free $\lambda\mu$ -calculus, *18th Conference Proceedings Japan Society for Software Science and Technology*, 2001.
- [Fuji02] K. Fujita: Continuation Semantics and CPS-Translation of $\lambda\mu$ -calculus, *Scientiae Mathematicae Japonicae*, 2002 to appear.
- [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.
- [LS86] J. Lambek and P. J. Scott: *Introduction to higher order categorical logic*, Cambridge University Press, 1986.
- [Pari92] M. Parigot: $\lambda\mu$ -Calculus: An Algorithmic Interpretation of Classical Natural Deduction, *Lecture Notes in Computer Science* 624, pp. 190–201, 1992.
- [Pari97] M. Parigot: Proofs of Strong Normalization for Second Order Classical Natural Deduction, *J. Symbolic Logic* 62 (4), pp. 1461–1479, 1997.
- [Scot72] D. Scott: Continuous Lattices, *Lecture Notes in Mathematics* 274, pp. 97–136, 1972.
- [Seli01] P. Selinger: Control Categories and Duality: on the Categorical Semantics of the Lambda-Mu Calculus, *Math. Struct. in Comp. Science* 11, pp. 207–260, 2001.

- [SR98] T. Streicher and B. Reus: Classical Logic, Continuation Semantics and Abstract Machines, *J. Functional Programming* 8, No. 6, pp. 543–572, 1998.
- [Stoy77] J. E. Stoy: *Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory*, The MIT Press, 1977.
- [SW96] A. Sabry and P. Wadler: A Reflection on Call-by-Value, *Proc. of the ACM SIG-PLAN International Conference on Functional Programming*, pp. 13–24, 1996.