Extensional Models of Untyped Lambda-mu Calculus

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

Shin-ya Katsumata
Research Institute for Mathematical Sciences, Kyoto University
sinya@kurims.kyoto-u.ac.jp

This paper proposes new mathematical models of the untyped Lambda-mu calculus. One is called the stream model, which is an extension of the lambda model, in which each term is interpreted as a function from streams to individual data. The other is called the stream combinatory algebra, which is an extension of the combinatory algebra, and it is proved that the extensional equality of the Lambda-mu calculus is equivalent to equality in stream combinatory algebras. In order to define the stream combinatory algebra, we introduce a combinatory calculus SCL, which is an abstraction-free system corresponding to the Lambda-mu calculus. Moreover, it is shown that stream models are algebraically characterized as a particular class of stream combinatory algebras.

1 Introduction

The $\lambda\mu$-calculus was originally proposed by Parigot in [8] as a term assignment system for the classical natural deduction, and some variants of $\lambda\mu$-calculus have been widely studied as typed calculi with control operators. Parigot noted that the μ-abstraction of the $\lambda\mu$-calculus can be seen as a potentially-infinite sequence of the λ-abstraction, and Saurin showed that an extension of the untyped $\lambda\mu$-calculus, which was originally considered by de Groote in [5] and was called $\Lambda\mu$-calculus by Saurin, can be seen as a stream calculus which enjoys some fundamental properties [9, 10, 11]. In particular, Saurin proved the separation theorem of the $\Lambda\mu$-calculus in [9], while it does not hold in the original $\lambda\mu$-calculus [4].

In [11], Saurin also proposed the Böhm-tree representation of the $\Lambda\mu$-terms. That suggests a relationship between the syntax and the semantics for the untyped $\Lambda\mu$-calculus like the neat correspondence between the Böhm-trees and Scott's D_∞ model for the untyped λ-calculus. However, models of the untyped $\Lambda\mu$-calculus have not been sufficiently studied yet, so we investigate how we can extend the results on the models of the λ-calculus to the $\Lambda\mu$-calculus.

In this paper, we give simple extensions of the λ-models and the combinatory algebras, and show that they can be seen as models of the untyped $\Lambda\mu$-calculus. First, we introduce *stream models* of the untyped $\Lambda\mu$-calculus, which are extended from the λ-models. The definition of stream model is based on the idea that the $\Lambda\mu$-calculus represents functions on streams, that is, in stream models, every $\Lambda\mu$-term is interpreted as a function from streams to individual data. Then, we give a new combinatory calculus SCL, which is an extension of the ordinary combinatory logic CL, and corresponds to the $\Lambda\mu$-calculus. The structure of SCL induces another model of the untyped $\Lambda\mu$-calculus, called *stream combinatory algebra*. We will show that the extensional equality of the $\Lambda\mu$-calculus is equivalent to equality in extensional stream combinatory algebras. We also show that the stream models are algebraically characterized as a particular class of the stream combinatory algebras.

2 Untyped $\Lambda\mu$-Calculus

First, we remind the untyped $\Lambda\mu$-calculus. We are following the notation of [9], because it is suitable to see the $\Lambda\mu$-calculus as a calculus operating streams.

Submitted to:
CL&C'12

Terms:

$$M,N ::= x \mid \lambda x.M \mid MN \mid \mu\alpha.M \mid M\alpha$$

Axioms:

$$\begin{aligned}
(\lambda x.M)N &=_{\beta_T} M[x := N] \\
(\mu\alpha.M)\beta &=_{\beta_S} M[\alpha := \beta] \\
\lambda x.Mx &=_{\eta_T} M && (x \notin FV(M)) \\
\mu\alpha.M\alpha &=_{\eta_S} M && (\alpha \notin FV(M)) \\
(\mu\alpha.M)N &=_{\mu} \mu\alpha.M[P\alpha := PN\alpha]
\end{aligned}$$

Figure 1: Untyped $\Lambda\mu$-calculus

Definition 2.1 ($\Lambda\mu$-calculus) Suppose that there are two disjoint sets of variables: one is the set Var_T of *term variables*, denoted by $x, y, \cdots$, and the other is the set Var_S of *stream variables*, denoted by $\alpha, \beta, \cdots$. Terms and axioms of the $\Lambda\mu$-calculus are given in the Fig. 1. The set of the $\Lambda\mu$-terms is denoted by $\mathrm{Term}_{\Lambda\mu}$. We use the following abbreviations: $\lambda x_1 x_2 \cdots x_n.M$ denotes $\lambda x_1.(\lambda x_2.(\cdots(\lambda x_n.M)\cdots))$ and similarly for μ, $MA_1 \cdots A_n$ denotes $(\cdots(MA_1)\cdots)A_n$, in which each A_i denotes either a term or a stream variable, and the top-level parentheses are also often omitted. Variable occurrences of x and α are bound in $\lambda x.M$ and $\mu\alpha.M$, respectively. Variable occurrences which are not bound are called free, and $FV(M)$ denotes the set of variables freely occurring in M. In the axioms, $M[x := N]$ and $M[\alpha := \beta]$ are the usual capture-avoiding substitutions, and $M[P\alpha := PN\alpha]$ recursively replaces each subterm of the form $P\alpha$ in M by $PN\alpha$. The relation $M =_{\Lambda\mu} N$ is the compatible equivalence relation defined from the axioms.

Contexts are defined as $K ::= []\alpha \mid K[]M]$, and $K[M]$ is defined in a usual way. The substitution $M[P\alpha := K[P]]$ recursively replaces each subterm of the form $P\alpha$ in M by $K[P]$.

Each context has the form $[]M_1 \cdots M_n\alpha$ and it corresponds to a stream data, the initial segment of which is $M_1 \cdots M_n$ and the rest is α. It is easy to see that $K[\mu\alpha.M] =_{\Lambda\mu} M[P\alpha := K[P]]$ for any term M and any context K.

The untyped $\Lambda\mu$-calculus can be seen as a calculus operating streams, in which the μ-abstractions represent functions on streams, and a term $MN_0 \cdots N_n\alpha$ means a function application of M to the stream data $[]N_0 \cdots N_n\alpha$. For example, the term $\mathsf{hd} = \lambda x.\mu\alpha.x$ is the function to get the head element of streams since we have $\mathsf{hd}N_0 \cdots N_n\beta =_{\beta_T} (\mu\alpha.N_0)N_1 \cdots N_n\beta =_{\mu} (\mu\alpha.N_0)\beta =_{\beta_S} N_0$. For another example, we have a term nth representing the function which takes a stream and a numeral c_n and returns the n-th element of the stream. The term nth is defined as

$$Y(\lambda f x.\mu\alpha.\lambda y.\mathsf{if}\ (\mathsf{zero?}\ y)\ \mathsf{then}\ x\ \mathsf{else}\ f\alpha(y-1)),$$

where Y is a fixed point operator in the λ-calculus, and we have

$$\mathsf{nth}N_0N_1N_2 \cdots N_n\beta\, c_i =_{\Lambda\mu} N_i$$

for any $0 \leq i \leq n$. However, the $\Lambda\mu$-calculus has no term representing a stream, and that means $\Lambda\mu$-terms do not directly represent any function which returns streams.

In Parigot's original $\lambda\mu$-calculus [8], terms of the form $P\alpha$, which are originally denoted by $[\alpha]P$, are distinguished as *named terms* from the ordinary terms, and bodies of μ-abstractions are restricted to the named terms. On the other hand, we consider $P\alpha$ as an ordinary term and any term can be the body

of μ-abstraction in the $\Lambda\mu$-calculus. For example, neither $M\alpha N$ nor $\mu\alpha.x$ is allowed as a term in the original $\lambda\mu$-calculus, whereas they are well-formed terms in the $\Lambda\mu$-calculus. Such extensions of the $\lambda\mu$-calculus in which the named terms are not distinguished have been originally studied by de Groote [5], and Saurin [9] considered a reduction system with the η-reduction, where another axiom

$$\mu\alpha.M \to_{fst} \lambda x.\mu\alpha.M[P\alpha := Px\alpha]$$

is chosen instead of (μ). For extensional equational systems, the axioms (μ) and (fst) are equivalent since

$$\mu\alpha.M =_{\eta_T} \lambda x.(\mu\alpha.M)x =_{\mu} \lambda x.\mu\alpha.M[P\alpha := Px\alpha], \text{ and}$$
$$(\mu\alpha.M)N =_{fst} (\lambda x.\mu\alpha.M[P\alpha := Px\alpha])N =_{\beta_T} \mu\alpha.M[P\alpha := PN\alpha]).$$

3 Stream Models

In this section, we introduce extensional stream models for the untyped $\Lambda\mu$-calculus. The definition follows the idea that the $\Lambda\mu$-terms represent functions on streams.

3.1 Definition of Extensional Stream Models

In the following, we use $\overline{\lambda}$ to represent meta-level functions. A *stream set* over a set D is a pair $(S,::)$ of a set S and a bijection $(::) : D \times S \to S$. A typical stream set over D is the $\mathbf{N}$-fold product of D, that is, $(D^{\mathbf{N}},::)$ where

$$d :: s = \overline{\lambda} n \in \mathbf{N}. \begin{cases} d & (n = 0) \\ s(n-1) & (n > 0). \end{cases}$$

For a function $f : D \times S \to E$, $\overline{\lambda} d :: s \in S.f(d,s)$ denotes the function $f \circ (::)^{-1} : S \to E$.

Definition 3.1 (Extensional stream models) An *extensional stream model* is a tuple $(D, S, [S \to D], ::, \Psi)$ such that

1. $(S,::)$ is a stream set over D.
2. $[S \to D]$ is a subset of $S \to D$.
3. $\Psi : [S \to D] \to D$ is a bijection. We write its inverse by Φ.
4. There is a (necessarily unique) function $[\![-]\!] : \mathrm{Term}_{\Lambda\mu} \times (\mathrm{Var}_T \to D) \times (\mathrm{Var}_S \to S) \to D$, called *meaning function*, such that

$$\begin{aligned} [\![x]\!]_{\rho,\theta} &= \rho(x) \\ [\![\lambda x.M]\!]_{\rho,\theta} &= \Psi(\overline{\lambda} d :: s \in S.\Phi([\![M]\!]_{\rho[x\mapsto d],\theta})(s)) \\ [\![MN]\!]_{\rho,\theta} &= \Psi(\overline{\lambda} s \in S.\Phi([\![M]\!]_{\rho,\theta})([\![N]\!]_{\rho,\theta} :: s)) \\ [\![\mu\alpha.M]\!]_{\rho,\theta} &= \Psi(\overline{\lambda} s \in S.[\![M]\!]_{\rho,\theta[\alpha\mapsto s]}) \\ [\![M\alpha]\!]_{\rho,\theta} &= \Phi([\![M]\!]_{\rho,\theta})(\theta(\alpha)). \end{aligned}$$

Here $\rho[x \mapsto d]$ is defined by

$$\rho[x \mapsto d](y) = \begin{cases} d & (x = y) \\ \rho(y) & (x \neq y), \end{cases}$$

and $\theta[\alpha \mapsto s]$ is defined similarly. We use the notation $d \star s$ to denote $\Phi(d)(s)$ for $d \in D$ and $s \in S$.

The condition 4 requires that each argument of Ψ is contained in $[S \to D]$. In the next subsection, we show that extensional stream models can be obtained from the solutions of the simultaneous recursive equations $D \times S \cong S$ and $S \Rightarrow D \cong D$ in a well-pointed CCC (Theorem 3.6).

Lemma 3.2 The following hold.

1. $[\![M[x := N]]\!]_{\rho,\theta} = [\![M]\!]_{\rho[x \mapsto [\![N]\!]_{\rho,\theta}],\theta}$.
2. $[\![M[\alpha := \beta]]\!]_{\rho,\theta} = [\![M]\!]_{\rho,\theta[\alpha \mapsto \theta(\beta)]}$.
3. $[\![M[P\alpha := PN\alpha]]\!]_{\rho,\theta} = [\![M]\!]_{\rho,\theta[\alpha \mapsto [\![N]\!]_{\rho,\theta} :: \theta(\alpha)]}$.

Proof. By induction on M. We show only the case of $M = M'\alpha$ for 3.

$$\begin{aligned}
&[\![(M'\alpha)[P\alpha := PN\alpha]]\!]_{\rho,\theta} \\
=&[\![M'[P\alpha := PN\alpha]N\alpha]\!]_{\rho,\theta} \\
=&[\![M'[P\alpha := PN\alpha]]\!]_{\rho,\theta} \star ([\![N]\!]_{\rho,\theta} :: \theta(\alpha)) \\
=&[\![M']\!]_{\rho,\theta[\alpha \mapsto [\![N]\!]_{\rho,\theta} :: \theta(\alpha)]} \star ([\![N]\!]_{\rho,\theta} :: \theta(\alpha)) && \text{(by IH)} \\
=&[\![M'\alpha]\!]_{\rho,\theta[\alpha \mapsto [\![N]\!]_{\rho,\theta} :: \theta(\alpha)]}.
\end{aligned}$$

□

Theorem 3.3 (Soundness) Let D be an arbitrary extensional stream model. If $M =_{\Lambda\mu} N$, then $[\![M]\!]_{\rho,\theta} = [\![N]\!]_{\rho,\theta}$ holds in D for any ρ and θ.

Proof. By induction on $M =_{\Lambda\mu} N$. We show only two cases, and the other cases are similarly proved by Lemma 3.2.

Case (β_T).

$$\begin{aligned}
[\![(\lambda x.M)N]\!]_{\rho,\theta} &= \Psi(\overline{\lambda}s.(\Psi(\overline{\lambda}d' :: s'.([\![M]\!]_{\rho[x \mapsto d'],\theta}) \star s')) \star ([\![N]\!]_{\rho,\theta} :: s)) \\
&= \Psi(\overline{\lambda}s.([\![M]\!]_{\rho[x \mapsto [\![N]\!]_{\rho,\theta}],\theta}) \star s) \\
&= [\![M]\!]_{\rho[x \mapsto [\![N]\!]_{\rho,\theta}],\theta} \\
&= [\![M[x := N]]\!]_{\rho,\theta} \qquad \text{(by Lemma 3.2.1)}
\end{aligned}$$

Case (μ).

$$\begin{aligned}
[\![(\mu\alpha.M)N]\!]_{\rho,\theta} &= \Psi(\overline{\lambda}s.(\Psi(\overline{\lambda}s'.[\![M]\!]_{\rho,\theta[\alpha \mapsto s']})) \star ([\![N]\!]_{\rho,\theta} :: s)) \\
&= \Psi(\overline{\lambda}s.[\![M]\!]_{\rho,\theta[\alpha \mapsto [\![N]\!]_{\rho,\theta} :: s]})
\end{aligned}$$

On the other hand, if we let $\theta' = \theta[\alpha \mapsto s]$, then the following holds.

$$\begin{aligned}
[\![\mu\alpha.M[P\alpha := PN\alpha]]\!]_{\rho,\theta} &= \Psi(\overline{\lambda}s.[\![M]\!]_{\rho,\theta'[\alpha \mapsto [\![N]\!]_{\rho,\theta'} :: s]}) \qquad \text{(by Lemma 3.2.3)} \\
&= \Psi(\overline{\lambda}s.[\![M]\!]_{\rho,\theta[\alpha \mapsto [\![N]\!]_{\rho,\theta} :: s]}) \qquad \text{(by } \alpha \notin FV(N))
\end{aligned}$$

□

Theorem 3.4 Every extensional stream model is an extensional λ-model in which the interpretation of λ-terms coincides with the interpretation in the stream model.

Proof. Let D be an extensional stream model, then we can define $[D \to D]$, $\Phi_0 : D \to [D \to D]$, and $\Psi_0 : [D \to D] \to D$ as follows.

$$\begin{aligned}
[D \to D] &:= \{f : D \to D \mid (\overline{\lambda} d :: s \in S.(f(d)) \star s) \in [S \to D]\} \\
\Phi_0(d) &:= \overline{\lambda} d' \in D.\Psi(\overline{\lambda} s \in S.d \star (d' :: s)) \\
\Psi_0(f) &:= \Psi(\overline{\lambda} d :: s \in S.(f(d)) \star s)
\end{aligned}$$

Note that these are variants of eval and abst in [15], and just based on the isomorphism $D \times S \simeq S$. Then, it is easily checked that D is a λ-model with Φ_0 and Ψ_0. The interpretation of the λ-terms in the λ-model, denoted $[\![\cdot]\!]^{\lambda}$ here, coincides with the interpretation in the stream model as follows:

$$\begin{aligned}
[\![\lambda x.M]\!]^{\lambda}_{\rho} &= \Psi_0(\overline{\lambda} d \in D.[\![M]\!]^{\lambda}_{\rho[x \mapsto d]}) \\
&= \Psi(\overline{\lambda} d' :: s' \in S.([\![M]\!]^{\lambda}_{\rho[x \mapsto d']}) \star s') && \text{(by Def. of } \Psi_0) \\
&= [\![\lambda x.M]\!]_{\rho} && \text{(by IH),} \\
[\![MN]\!]^{\lambda}_{\rho} &= \Phi_0([\![M]\!]^{\lambda}_{\rho})([\![N]\!]^{\lambda}_{\rho}) \\
&= \Psi(\overline{\lambda} s \in S.([\![M]\!]^{\lambda}_{\rho}) \star ([\![N]\!]^{\lambda}_{\rho} :: s)) && \text{(by Def. of } \Phi_0) \\
&= [\![MN]\!]_{\rho} && \text{(by IH).}
\end{aligned}$$

□

3.2 Categorical Stream Models

In a categorical setting, a solution (D,S) of the following simultaneous recursive equations in a CCC provides a model of the $\Lambda\mu$-calculus.

$$D \times S \simeq S, \quad S \Rightarrow D \simeq D \tag{1}$$

Definition 3.5 (Categorical stream models) A *categorical stream model* in a CCC $\mathbf{C}$ is a tuple (D,S,c,ψ) of objects D and S, and isomorphisms $c : D \times S \to S$ and $\psi : S \Rightarrow D \to D$.

When $\mathbf{C}$ has countable products, the solutions of the following recursive equation:

$$D^{\mathbf{N}} \Rightarrow D \simeq D \tag{2}$$

yield categorical stream models, as we always have $D^{\mathbf{N}} \simeq D \times D^{\mathbf{N}}$.

Given a categorical stream model (D,S,c,ψ), we can interpret $\Lambda\mu$-terms as a morphism $[\![M]\!]_{\vec{x},\vec{\alpha}} : D^{|\vec{x}|} \times S^{|\vec{\alpha}|} \to D$, where $\vec{x}$ (resp. $\vec{\alpha}$) is a finite sequence of distinct term (stream) variables such that every free term (stream) variable in M occurs in $\vec{x}$ ($\vec{\alpha}$), and $|\vec{x}|$ ($|\vec{\alpha}|$) is the length of $\vec{x}$ ($\vec{\alpha}$). We omit the details of this interpretation, as it is a straightforward categorical formulation of the meaning function in Definition 3.1.

When the underlying CCC $\mathbf{C}$ of a categorical stream model is well-pointed (that is, the global element functor $\mathbf{C}(1,-) : \mathbf{C} \to \mathrm{Set}$ is faithful), we can convert it to an extensional stream model.

Theorem 3.6 Let $\mathbf{C}$ be a well-pointed CCC. For any categorical stream model (D,S,c,ψ) in $\mathbf{C}$, the following tuple is an extensional stream model:

$$(\mathbf{C}(1,D),\ \mathbf{C}(1,S),\ \{\mathbf{C}(1,f) \mid f \in \mathbf{C}(S,D)\},\ \overline{\lambda}(f,g).c \circ \langle f,g \rangle,\ \Psi),$$

where Ψ is the function defined by $\Psi(\mathbf{C}(1,f)) = \psi \circ \lambda(f \circ \pi_2)$.

For instance, in the well-pointed CCC of pointed CPOs and all continuous functions, the standard inverse limit method [13, 14] applied to the following embedding-projection pair ($e : D_0 \to D_0^{\mathbf{N}} \Rightarrow D_0, p : D_0^{\mathbf{N}} \Rightarrow D_0 \to D_0$):

$$e(x) = \overline{\lambda} y \in D_0^{\mathbf{N}}.x, \quad p(f) = f(\bot, \ldots)$$

on a pointed CPO D_0 containing at least two elements yields a non-trivial solution of (2). From this solution, an extensional stream model is derived by Theorem 3.6. This model distinguishes $[\![\lambda xy.x]\!]$ and $[\![\lambda xy.y]\!]$, hence, we obtain a model theoretic consistency proof of the $\Lambda\mu$-calculus (consistency also follows from confluence, which has been proved in [12]).

4 Stream Combinatory Algebra

We give another model of the untyped $\Lambda\mu$-calculus. It is called stream combinatory algebra, which is an extension of the combinatory algebra corresponding to the combinatory logic CL.

4.1 Combinatory Calculus SCL

We introduce a new combinatory calculus SCL, and show that SCL is equivalent to the $\Lambda\mu$-calculus. This result is an extension of the equivalence between the λ-calculus and the untyped variant of the ordinary combinatory logic CL with the combinators K and S. In SCL, the combinators K and S are denoted by K_0 and S_0, respectively.

Definition 4.1 (SCL) Similarly to the $\Lambda\mu$-calculus, SCL has two sorts of variables: term variables Var_T and stream variables Var_S. Constants, terms, streams, axioms, and extensionality rules of SCL are given in Fig. 2. The set of the SCL-terms and the set of the SCL-streams are denoted by $\mathrm{Terms}_{\mathsf{SCL}}$ and $\mathrm{Streams}_{\mathsf{SCL}}$, respectively. The set of variables occurring in T is denoted by $FV(T)$. We suppose that the binary function symbols $(\cdot)$ and $(\star)$ have the same associative strength, and both are left associative. For example, $T_1 \cdot T_2 \star \mathscr{S}_3 \cdot T_4$ denotes $((T_1 \cdot T_2) \star \mathscr{S}_3) \cdot T_4$. The substitutions $T[x := T']$ and $T[\alpha := \mathscr{S}]$ are defined straightforwardly. The relation $T =_{\mathsf{SCL}} U$ is the compatible equivalence relation defined from the axioms and the extensionality rules.

The new operation $(\star)$ represents the function application for streams, which corresponds to the application $M\alpha$ in the $\Lambda\mu$-calculus.

In the following, we think that the term of the form $T_1 \cdot T_2 \star \mathscr{S}_3$ is simpler than $T_1 \star (T_2 :: \mathscr{S}_3)$, and that is formalized as the following measure $|T|$.

Definition 4.2 The measure $|T|$ of SCL-terms is defined as $|T| = \mathsf{c}(T) + \mathsf{m}(T)$, where $\mathsf{c}(T)$ is the number of the symbol :: occurring in T, and $\mathsf{m}(T)$ is the number of nodes of the syntax tree of T.

It is easily seen that if T is a subterm of U then $|T| < |U|$, and $|T_1 \cdot T_2 \star \mathscr{S}_3| < |T_1 \star (T_2 :: \mathscr{S}_3)|$, which follows from $\mathsf{m}(T_1 \cdot T_2 \star \mathscr{S}_3) = \mathsf{m}(T_1 \star (T_2 :: \mathscr{S}_3))$.

The $\Lambda\mu$-calculus and SCL are equivalent through the following translations.

Definition 4.3 (Translations between $\Lambda\mu$ and SCL) 1. For $T \in \mathrm{Terms}_{\mathsf{SCL}}$ and $x \in \mathrm{Var}_T$, we define the SCL-term $\lambda^* x.T$ inductively on $|T|$ as follows:

$$\begin{aligned}
\lambda^* x.x &= \mathsf{S}_0 \cdot \mathsf{K}_0 \cdot \mathsf{K}_0 \\
\lambda^* x.T &= \mathsf{K}_0 \cdot T && (x \notin FV(T)) \\
\lambda^* x.(T \cdot U) &= \mathsf{S}_0 \cdot (\lambda^* x.T) \cdot (\lambda^* x.U) \\
\lambda^* x.(T \star \alpha) &= \mathsf{C}_{10} \cdot (\lambda^* x.T) \star \alpha \\
\lambda^* x.(T \star (U :: \alpha)) &= \lambda^* x.(T \cdot U \star \alpha).
\end{aligned}$$

Constants:

$$C ::= \mathsf{K}_0 \mid \mathsf{K}_1 \mid \mathsf{S}_0 \mid \mathsf{S}_1 \mid \mathsf{C}_{10} \mid \mathsf{C}_{11} \mid \mathsf{W}_1$$

Terms:

$$T, U ::= C \mid x \mid T \cdot U \mid T \star \mathscr{S}$$

Streams:

$$\mathscr{S} ::= \alpha \mid T :: \mathscr{S}$$

Axioms:

$$\begin{aligned} \mathsf{K}_0 \cdot T_1 \cdot T_2 &= T_1 & \mathsf{K}_1 \cdot T_1 \star \mathscr{S}_2 &= T_1 \\ \mathsf{S}_0 \cdot T_1 \cdot T_2 \cdot T_3 &= T_1 \cdot T_3 \cdot (T_2 \cdot T_3) & \mathsf{S}_1 \cdot T_1 \cdot T_2 \star \mathscr{S}_3 &= T_1 \star \mathscr{S}_3 \cdot (T_2 \star \mathscr{S}_3) \\ \mathsf{C}_{10} \cdot T_1 \star \mathscr{S}_2 \cdot T_3 &= T_1 \cdot T_3 \star \mathscr{S}_2 & \mathsf{C}_{11} \cdot T_1 \star \mathscr{S}_2 \star \mathscr{S}_3 &= T_1 \star \mathscr{S}_3 \star \mathscr{S}_2 \\ \mathsf{W}_1 \cdot T_1 \star \mathscr{S}_2 &= T_1 \star \mathscr{S}_2 \star \mathscr{S}_2 & T_1 \star (T_2 :: \mathscr{S}_3) &= T_1 \cdot T_2 \star \mathscr{S}_3 \end{aligned}$$

Extensionality rules:

$$\frac{T \cdot x = U \cdot x \quad x \notin FV(T) \cup FV(U)}{T = U}\ (\zeta_T) \qquad \frac{T \star \alpha = U \star \alpha \quad \alpha \notin FV(T) \cup FV(U)}{T = U}\ (\zeta_S)$$

Figure 2: SCL

For $T \in \mathrm{Term}_{\mathsf{SCL}}$ and $\alpha \in \mathrm{Var}_S$, we define the SCL-term $\mu^* \alpha.T$ inductively on $|T|$ as follows:

$$\begin{aligned} \mu^* \alpha.T &= \mathsf{K}_1 \cdot T && (\alpha \notin FV(T)) \\ \mu^* \alpha.(T \cdot U) &= \mathsf{S}_1 \cdot (\mu^* \alpha.T) \cdot (\mu^* \alpha.U) \\ \mu^* \alpha.(T \star \alpha) &= \mathsf{W}_1 \cdot (\mu^* \alpha.T) \\ \mu^* \alpha.(T \star \beta) &= \mathsf{C}_{11} \cdot (\mu^* \alpha.T) \star \beta && (\alpha \neq \beta) \\ \mu^* \alpha.(T \star (U :: \alpha)) &= \mu^* \alpha.(T \cdot U \star \alpha). \end{aligned}$$

Then the mapping M^* from $\mathrm{Term}_{\Lambda\mu}$ to $\mathrm{Term}_{\mathsf{SCL}}$ is defined by

$$\begin{aligned} x^* &= x \\ (\lambda x.M)^* &= \lambda^* x.M^* & (MN)^* &= M^* \cdot N^* \\ (\mu \alpha.M)^* &= \mu^* \alpha.M^* & (M\alpha)^* &= M^* \star \alpha. \end{aligned}$$

2. The mappings T_* from $\mathrm{Term}_{\mathsf{SCL}}$ to $\mathrm{Term}_{\lambda\mu}$ and $\mathscr{S}_*$ from $\mathrm{Stream}_{\mathsf{SCL}}$ to contexts are defined by

$$\begin{aligned} (\mathsf{K}_0)_* &= \lambda xy.x & x_* &= x \\ (\mathsf{K}_1)_* &= \lambda x.\mu \alpha.x & (T \cdot U)_* &= T_* U_* \\ (\mathsf{S}_0)_* &= \lambda xyz.xz(yz) & (T \star \mathscr{S})_* &= \mathscr{S}_*[T_*] \\ (\mathsf{S}_1)_* &= \lambda xy.\mu \alpha.x\alpha(y\alpha) \\ (\mathsf{C}_{10})_* &= \lambda x.\mu \alpha.\lambda y.xy\alpha & \alpha_* &= []\alpha \\ (\mathsf{C}_{11})_* &= \lambda x.\mu \alpha\beta.x\beta\alpha & (T :: \mathscr{S})_* &= \mathscr{S}_*[[]T_*]. \\ (\mathsf{W}_1)_* &= \lambda x.\mu \alpha.x\alpha\alpha \end{aligned}$$

By the extensionality of SCL, the definitions of $\lambda^* x.T$ and $\mu^* \alpha.T$ such that 1 of the following lemma holds are unique modulo $=_{\mathsf{SCL}}$.

Lemma 4.4 The following hold.
1. $(\lambda^* x.T) \cdot U =_{\mathsf{SCL}} T[x := U]$ and $(\mu^* \alpha.T) \star \mathscr{S} =_{\mathsf{SCL}} T[\alpha := \mathscr{S}]$.
2. If $T =_{\mathsf{SCL}} U$, then $\lambda^* x.T =_{\mathsf{SCL}} \lambda^* x.U$ and $\mu^* \alpha.T =_{\mathsf{SCL}} \mu^* \alpha.U$.

Proof. 1. By induction on $|T|$.

2. By 1, we have $(\lambda^* x.T) \cdot x =_{\mathsf{SCL}} T$ and $(\lambda^* x.U) \cdot x =_{\mathsf{SCL}} U$. Since $T =_{\mathsf{SCL}} U$, we have $(\lambda^* x.T) \cdot x =_{\mathsf{SCL}} (\lambda^* x.U) \cdot x$, and hence $\lambda^* x.T =_{\mathsf{SCL}} \lambda^* x.U$ by (ζ_T). □

Lemma 4.5 The following hold.
1. $(M[x := N])^* =_{\mathsf{SCL}} M^*[x := N^*]$.
2. $(M[\alpha := \beta])^* =_{\mathsf{SCL}} M^*[\alpha := \beta]$.
3. $(M[P\alpha := PN\alpha])^* =_{\mathsf{SCL}} M^*[\alpha := N^* :: \alpha]$.

Proof. By induction on M. We show only the case of $M = \lambda y.M'$ for 1. We suppose that $y \notin FV(N)$ and $y \not\equiv x$ by renaming bound variables. We have $((\lambda y.M')[x := N])^* \cdot y = (\lambda y.M'[x := N])^* \cdot y = (\lambda^* y.(M'[x := N])^*) \cdot y =_{\mathsf{SCL}} (M'[x := N])^*$ by Lemma 4.4.1, and it is identical with $M'^*[x := N^*]$ by the induction hypothesis. On the other hand, we have $(\lambda^* y.M'^*[x := N^*]) \cdot y =_{\mathsf{SCL}} M'^*[x := N^*]$. Hence, by (ζ_T), we have $((\lambda y.M')[x := N])^* =_{\mathsf{SCL}} (\lambda y.M')^*[x := N^*]$. □

Lemma 4.6 The following hold.
1. $M =_{\Lambda\mu} N$ implies $M^* =_{\mathsf{SCL}} N^*$.
2. $T =_{\mathsf{SCL}} U$ implies $T_* =_{\Lambda\mu} U_*$.
3. $(M^*)_* =_{\Lambda\mu} M$.
4. $(T_*)^* =_{\mathsf{SCL}} T$ and $(\mathscr{S}_*[M])^* =_{\mathsf{SCL}} M^* \star \mathscr{S}$

Proof. By the previous lemmas, they are proved by induction straightforwardly. □

It is shown that the combinatory calculus SCL is equivalent to the $\Lambda\mu$-calculus in the following sense.

Theorem 4.7 1. For any $\Lambda\mu$-terms M and N, $M =_{\Lambda\mu} N$ iff $M^* =_{\mathsf{SCL}} N^*$.

2. For any SCL-terms T and U, $T =_{\mathsf{SCL}} U$ iff $T_* =_{\Lambda\mu} U_*$.

Proof. 1. The only-if part is Lemma 4.6.1, and the if part is proved by Lemma 4.6.2 and 4.6.3 as $M =_{\Lambda\mu} (M^*)_* =_{\Lambda\mu} (N^*)_* =_{\Lambda\mu} N$.

2. Similar to 1 by 1, 2, and 4 of Lemma 4.6. □

4.2 Stream Combinatory Algebra

The stream combinatory algebras are given as models of SCL. Since SCL is equivalent to the $\Lambda\mu$-calculus in the sense of Theorem 4.7, they are also models of the untyped $\Lambda\mu$-calculus.

Definition 4.8 (Stream combinatory algebras) (1) For non-empty sets D and S, a tuple $(D, S, \cdot, \star, ::)$ is called a *stream applicative structure* if $(\cdot) : D \times D \to D$, $(\star) : D \times S \to D$, and $(::) : D \times S \to S$ are mappings such that

$$d_1 \star (d_2 :: s_3) = d_1 \cdot d_2 \star s_3$$

for any $d_1, d_2 \in D$ and $s_3 \in S$.

(2) A stream applicative structure D is *extensional* if the following hold for any $d, d' \in D$:

$$\forall d_0 \in D[d \cdot d_0 = d' \cdot d_0] \text{ implies } d = d',$$
$$\forall s_0 \in S[d \star s_0 = d' \star s_0] \text{ implies } d = d'.$$

(3) A stream applicative structure D is called a *stream combinatory algebra* if D contains distinguished elements k_0, k_1, s_0, s_1, c_{10}, c_{11}, and w_1 such that the following hold for any $d_1, d_2, d_3 \in D$ and $s_2, s_3 \in S$.

$$\mathsf{k}_0 \cdot d_1 \cdot d_2 = d_1 \qquad \mathsf{k}_1 \cdot d_1 \star s_2 = d_1$$
$$\mathsf{s}_0 \cdot d_1 \cdot d_2 \cdot d_3 = d_1 \cdot d_3 \cdot (d_2 \cdot d_3) \qquad \mathsf{s}_1 \cdot d_1 \cdot d_2 \star s_3 = d_1 \star s_3 \cdot (d_2 \star s_3)$$
$$\mathsf{c}_{10} \cdot d_1 \star s_2 \cdot d_3 = d_1 \cdot d_3 \star s_2 \qquad \mathsf{c}_{11} \cdot d_1 \star s_2 \star s_3 = d_1 \star s_3 \star s_2$$
$$\mathsf{w}_1 \cdot d_1 \star s_2 = d_1 \star s_2 \star s_2$$

Note that, for a stream applicative structure $(D, S, \cdot, \star, ::)$, the set S is not necessarily a stream set on D in the sense of Section 3, and we will call D *standard* if $(S, ::)$ is a stream set on D.

It is clear that any stream combinatory algebra is always a combinatory algebra by ignoring the stream part, that is, $(\star)$, $(::)$, k_1, s_1, c_{10}, c_{11}, and w_1. Therefore, any extensional stream combinatory algebra is an extensional combinatory algebra, and hence an extensional λ-model.

We can interpret SCL in stream combinatory algebras in a straightforward way.

Definition 4.9 (Interpretation of SCL) Let $(D, S, \cdot, \star, ::)$ be a stream combinatory algebra. The *meaning functions* $\langle\!|-|\!\rangle^T : \mathrm{Term}_{\mathsf{SCL}} \times (\mathrm{Var}_T \to D) \times (\mathrm{Var}_S \to S) \to D$ and $\langle\!|-|\!\rangle^S : \mathrm{Stream}_{\mathsf{SCL}} \times (\mathrm{Var}_T \to D) \times (\mathrm{Var}_S \to S) \to S$ are defined by:

$$\langle\!|C|\!\rangle^T_{\rho,\theta} = c \qquad \langle\!|\alpha|\!\rangle^S_{\rho,\theta} = \theta(\alpha)$$
$$\langle\!|x|\!\rangle^T_{\rho,\theta} = \rho(x) \qquad \langle\!|T :: \mathscr{S}|\!\rangle^S_{\rho,\theta} = \langle\!|T|\!\rangle^T_{\rho,\theta} :: \langle\!|\mathscr{S}|\!\rangle^S_{\rho,\theta},$$
$$\langle\!|T \cdot U|\!\rangle^T_{\rho,\theta} = \langle\!|T|\!\rangle^T_{\rho,\theta} \cdot \langle\!|U|\!\rangle^T_{\rho,\theta}$$
$$\langle\!|T \star \mathscr{S}|\!\rangle^T_{\rho,\theta} = \langle\!|T|\!\rangle^T_{\rho,\theta} \star \langle\!|\mathscr{S}|\!\rangle^S_{\rho,\theta}$$

where c denotes the element of D corresponding to the constant C, that is, $\langle\!|\mathsf{K}_0|\!\rangle^T_{\rho,\theta} = \mathsf{k}_0$, $\langle\!|\mathsf{S}_0|\!\rangle^T_{\rho,\theta} = \mathsf{s}_0$, and so on. We often omit the superscript T or S.

Theorem 4.10 (Soundness and completeness) For any SCL-terms T and U, $T =_{\mathsf{SCL}} U$ iff $\langle\!|T|\!\rangle_{\rho,\theta} = \langle\!|U|\!\rangle_{\rho,\theta}$ in any extensional stream combinatory algebra for any ρ and θ.

Proof. (Only-if part) The soundness can be proved by straightforward induction on $T =_{\mathsf{SCL}} U$.

(If part) We can construct a term model as follows. Let $D = \mathrm{Term}_{\mathsf{SCL}} / =_{\mathsf{SCL}}$ and $S = \mathrm{Stream}_{\mathsf{SCL}} / =_{\mathsf{SCL}}$, and the equivalence classes in D and S are denoted such as $[T]$ and $[\mathscr{S}]$. The operations are defined as $[T] \cdot [U] = [T \cdot U]$, $[T] \star [\mathscr{S}] = [T \star \mathscr{S}]$, and $[T] :: [\mathscr{S}] = [T :: \mathscr{S}]$. The element k_0 is defined as $[\mathsf{K}_0]$ and similar for the other constants. The resulting structure is easily proved to be an extensional stream combinatory algebra. If we take ρ and θ as $\rho(x) = [x]$ and $\theta(\alpha) = [\alpha]$, respectively, then $\langle\!|T|\!\rangle_{\rho,\theta} = [T]$ for any $T \in \mathrm{Term}_{\mathsf{SCL}}$, hence we have that $\langle\!|T|\!\rangle_{\rho,\theta} = \langle\!|U|\!\rangle_{\rho,\theta}$ implies $T =_{\mathsf{SCL}} U$. □

Corollary 4.11 For any $\Lambda\mu$-terms M and N, $M =_{\Lambda\mu} N$ iff $\langle\!|M^*|\!\rangle_{\rho,\theta} = \langle\!|N^*|\!\rangle_{\rho,\theta}$ in any extensional stream combinatory algebra for any ρ and θ.

Proof. It immediately follows from Theorem 4.7 and Theorem 4.10. □

5 Algebraic Characterization of Stream Models

Definition 3.1 of the extensional stream models is a direct one, but it depends on the definability of the meaning function on the $\Lambda\mu$-terms. In this section, we give a syntax-free characterization for the extensional stream models, that is, the class of the extensional stream models exactly coincides with the subclass of the extensional stream combinatory algebras in which S is a stream set on D.

Definition 5.1 A stream applicative structure $(D, S, \cdot, \star, ::)$ is *standard* if $(S, ::)$ is a stream set on D.

Note that, for standard stream applicative structures, the extensionality for term application $(\cdot)$ follows from the extensionality for $(\star)$ since $(::)$ is surjective: suppose $d_1 \cdot d = d_2 \cdot d$ for any $d \in D$, then for any $s \in S$ we have $d_1 \cdot d \star s = d_2 \cdot d \star s$, which means $d_1 \star (d :: s) = d_2 \star (d :: s)$ for any d and s. Hence $d_1 = d_2$ by the extensionality with respect to $\star$.

Theorem 5.2 For a non-empty set D and a stream set $(S, ::)$ on D, the following are equivalent.

1. (D, S) is an extensional stream model with some $[S \to D]$ and Ψ.
2. (D, S) is a standard extensional combinatory algebra with some operations $(\cdot)$ and $(\star)$, and some elements k_0, k_1, s_0, s_1, c_{10}, c_{11}, w_1 in D.

Proof. (1$\Longrightarrow$2) Suppose $(D, S, [S \to D], ::, \Psi)$ is an extensional stream model. Define

$$d \star s = \Phi(d)(s) \qquad\qquad d \cdot d' = \Psi(\overline{\lambda} s \in S.\Phi(d)(d' :: s)),$$

where we should note that $d \cdot d'$ is identical to $[\![xy]\!]_{\rho[x \mapsto d, y \mapsto d']}$ and hence it is always defined. Define $\mathsf{k}_0 = [\![\lambda xy.x]\!]$ and so on. Then $(D, S, \cdot, \star, ::)$ is a standard extensional stream combinatory algebra. Indeed, it is a stream applicative structure, since

$$d_1 \cdot d_2 \star s_3 = \Phi(\Psi(\overline{\lambda} s.\Phi(d_1)(d_2 :: s)))(s_3) = \Phi(d_1)(d_2 :: s_3) = d_1 \star (d_2 :: s_3).$$

(2$\Longrightarrow$1) Suppose $(D, S, \cdot, \star, ::)$ is a standard extensional stream combinatory algebra. Define $[S \to D] := \{f_d \mid d \in D\}$, where f_d denotes $\overline{\lambda} s \in S.d \star s$. Then $\Phi(d) = f_d$ and $\Psi(f_d) = d$ are well-defined since D is extensional, and they give a bijection between $[S \to D]$ and D. We can see that the interpretation $[\![M]\!]_{\rho,\theta}$ with respect to Φ and Ψ coincides with $\langle\!|M^*|\!\rangle_{\rho,\theta}$. That is shown by the following lemmas for any SCL-term T:

$$\langle\!|\lambda^* x.T|\!\rangle_{\rho,\theta} \cdot d = \langle\!|T|\!\rangle_{\rho[x \mapsto d],\theta} \qquad\qquad \langle\!|\mu^* \alpha.T|\!\rangle_{\rho,\theta} \star s = \langle\!|T|\!\rangle_{\rho,\theta[\alpha \mapsto s]}.$$

In the case of $M = \lambda x.N$, $[\![M]\!]_{\rho,\theta} = \langle\!|M^*|\!\rangle_{\rho,\theta}$ is proved as follows.

$$\begin{aligned}
\langle\!|M^*|\!\rangle_{\rho,\theta} \star (d :: s) &= \langle\!|M^*|\!\rangle_{\rho,\theta} \cdot d \star s \\
&= \langle\!|N^*|\!\rangle_{\rho[x \mapsto d],\theta} \star s && \text{(by the lemma)} \\
&= [\![N]\!]_{\rho[x \mapsto d],\theta} \star s && \text{(by IH)} \\
&= \Phi([\![N]\!]_{\rho[x \mapsto d],\theta})(s)
\end{aligned}$$

Therefore we have $\overline{\lambda} d :: s.\Phi([\![N]\!]_{\rho[x \mapsto d],\theta})(s) = \overline{\lambda} d :: s.\langle\!|M^*|\!\rangle_{\rho,\theta} \star (d :: s) = f_{\langle\!|M^*|\!\rangle_{\rho,\theta}} \in [S \to D]$, and hence $[\![M]\!]_{\rho,\theta}$ is defined and identical to $\Psi(f_{\langle\!|M^*|\!\rangle_{\rho,\theta}}) = \langle\!|M^*|\!\rangle_{\rho,\theta}$. The other cases are similarly proved. Hence, $(D, S, [S \to D], ::, \Psi)$ is an extensional stream model. □

6 Conclusion

We have proposed models of the untyped $\Lambda\mu$-calculus: the set-theoretic and the categorical stream models, and the stream combinatory algebras. We have also shown that extensional stream models are algebraically characterized as a particular class of the extensional stream combinatory algebras. The stream combinatory algebra has been induced from the new combinatory calculus SCL, which exactly corresponds to the untyped $\Lambda\mu$-calculus.

6.1 Related Work

Models of the untyped $\lambda\mu$-calculus. In [15], Streicher and Reus proposed the continuation models for the untyped $\lambda\mu$-calculus (which is a variant of Parigot's original $\lambda\mu$-calculus) based on the idea that the $\lambda\mu$-calculus is a calculus of continuations. If we see each stream $d :: s$ as a pair (d,s) of a function argument d and a continuation s, the meaning function for the stream models looks exactly the same as that for the continuation models.

In the untyped $\lambda\mu$-calculus in [15], the named terms are distinguished from the ordinary terms. In the continuation models, an object R (called *response object*) for the denotations of named terms is fixed first, then the object D for the denotations of the ordinary terms and the object S for continuations are respectively given as the solutions of the following simultaneous recursive equations:

$$D \times S \cong S, \quad S \Rightarrow R \cong D. \tag{3}$$

These equations say that the continuations are streams of ordinary terms, and the ordinary terms can act as functions from continuations to responses (i.e. results of computations). On the other hand, in the $\Lambda\mu$-calculus, the named terms and terms are integrated into one syntactic category, thus allowing us to pass terms to named terms, such as $M\alpha N$. In the model side, this extension corresponds to that the response object R in (3) is replaced by D, resulting in the simultaneous recursive equations (1).

In [16], van Bakel et al. considered intersection type systems and filter models for the $\lambda\mu$-calculus based on the idea of the continuation models of Streicher and Reus. They considered only the original $\lambda\mu$-calculus, and $\Lambda\mu$-terms such as $\mu\alpha.x$ have no type except for ω in the proposed intersection type system, and hence, they are interpreted as the bottom element in the filter model. They also showed that every continuation model can be a model of the $\Lambda\mu$-calculus. The idea is to translate each $\Lambda\mu$-term to a $\lambda\mu$-term as $\mu\alpha.M$ to $\mu\alpha.M\alpha$ and $M\alpha$ to $\mu\beta.M\alpha$ with a fresh β. However, as pointed out in [16], the axiom (β_S) is unsound for this interpretation in general, whereas it is sound in our stream models.

Akama [1] showed that the untyped $\lambda\mu$-calculus can be interpreted in partial combinatory algebras. It is based on the idea that μ-abstractions are functions on streams. However, it restricts terms to affine ones, that is, each bound variable must not occur more than once.

Fujita [6] considered a reduction system for the $\lambda\mu$-calculus with (β_T), (η_T), (μ), and (fst) rules, and gives a translation from the $\lambda\mu$-calculus to the λ-calculus which preserves the equality, and hence it is shown that any extensional λ-model is a model of the $\lambda\mu$-calculus. In the translation, each μ-abstraction is interpreted as a potentially infinite λ-abstraction by means of a fixed point operator. However, it considers neither (β_S) nor (η_S), and it seems hard to obtain a similar result for them.

Combinatory logic and classical logic. Baba et al. considered some extensions of the λ-calculus with combinators corresponding to classical axioms such as Peirce's law and double negation elimination in [2].

Nour [7] introduced the classical combinatory logic corresponding to Barbanera and Berardi's symmetric λ-calculus [3]. The classical combinatory logic has two kinds of application operators: one is the

ordinary function application, and the other represents the interaction of terms and continuations, which is based on the same idea as the stream application operator in SCL (and denoted by the same symbol $\star$). Nour's classical combinatory logic is a typed calculus corresponding to classical logic, and its weak reduction corresponds to the reduction of the symmetric λ-calculus. On the other hand, we have not found any reasonable type system for SCL as discussed below, but SCL corresponds to the $\Lambda\mu$-calculus, and, in particular, it can represent the μ-abstraction over continuation variables.

6.2 Further Study

(Extensional) stream models. One natural direction of study is to analyze the local structure of the domain-theoretic extensional stream models constructed from the solutions of (2) in Section 3.2. How do they relate to the Böhm-tree representation proposed in [11]? Do these models enjoy the approximation theorem? Which syntactic equality corresponds to the equality in these models?

We have considered only extensional theories and models in this paper. We can naïvely define non-extensional stream models by weakening the condition $[S \to D] \simeq D$ to $[S \to D] \lhd D$, and then the functions Φ_0 and Ψ_0 in Theorem 3.4 are still well-defined. However, under such a structure, we always have $\Psi_0 \circ \Phi_0 = \mathrm{id}$, so the extensionality axiom η_T is unexpectedly sound, for example $[\![\lambda xy.xy]\!] = [\![\lambda x.x]\!]$ always holds. Furthermore, we do not know how to derive that $\Phi_0 \circ \Psi_0 = \mathrm{id}$, which is essential for modeling the β-equality of the term application. It is future work to study how we can define appropriate notion of the models of the non-extensional $\Lambda\mu$-calculus.

Moreover, syntactic correspondence between non-extensional theories of the $\Lambda\mu$-calculus and SCL is still unclear and it is future work to study on it.

Types and classical logic. The $\lambda\mu$-calculus was originally introduced as a typed calculus corresponding to the classical natural deduction in the sense of the Curry-Howard isomorphism. It is future work to adapt our discussion to a typed setting and to study the relationship to classical logic. It is well-known that the combinatory logic with types exactly corresponds to the Hilbert-style proof system of intuitionistic logic. On the other hand, it is unclear how we can consider SCL as a typed calculus, since the $\Lambda\mu$-terms corresponding to the constants of SCL are not typable in the ordinary typed $\lambda\mu$-calculus, for example, $(\mathsf{S}_1)_* = \lambda xy.\mu\alpha.x\alpha(y\alpha)$.

Acknowledgments We are grateful to Dana Scott, Kazushige Terui, Makoto Tatsuta, and anonymous reviewers for helpful comments, and to Daisuke Kimura for fruitful discussions.

References

[1] Akama, Y. Limiting partial combinatory algebras towards infinitary. In *Proceedings of Computer Science Logic (CSL 2001)*, volume 2142 of *LNCS*, pages 399–414, 2001.

[2] Baba, K., Kameyama, Y., and Hirokawa, S. Combinatory logic and λ-calculus for classical logic. *Bulletin of Informatics and Cybernetics*, 32:105–122, 2000.

[3] Barbanera, F. and Berardi, S. A symmetric lambda-calculus for classical program extraction. In *TACS'94*, pages 495–515, 1994.

[4] David, R. and Py, W. $\lambda\mu$-calculus and Böhm's theorem. *The Journal of Symbolic Logic*, 66:407–413, 2001.

[5] de Groote, P. On the relation between the $\lambda\mu$-calculus and the syntactic theory of sequential control. In F. Pfenning, editor, *Proceedings of the International Conference on Logic Programming and Automated Reasoning (LPAR'94)*, volume 822 of *LNCS*, pages 31–43, 1994.

[6] Fujita, K. An interpretation of $\lambda\mu$-calculus in λ-calculus. *Information Processing Letters*, 84:261–264, 2002.

[7] Nour, K. Classical combinatory logic. In *Computational Logic and Applications (CLA'05)*, DMTCS proceedings, pages 87–96, 2006.

[8] Parigot, M. $\lambda\mu$-calculus: an algorithmic interpretation of classical natural deduction. In *Proceedings of the International Conference on Logic Programming and Automated Reasoning (LPAR '92)*, volume 624 of *LNCS*, pages 190–201, 1992.

[9] Saurin, A. Separation with streams in the $\Lambda\mu$-calculus. In *20th Annual IEEE Symposium on Logic in Computer Science (LICS' 05)*, pages 356–365, 2005.

[10] Saurin, A. A hierarchy for delimited control in call-by-name. In *13th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS 2010)*, volume 6014 of *LNCS*, pages 374–388, 2010.

[11] Saurin, A. Standardization and Böhm trees for $\Lambda\mu$-calculus. In *Tenth International Symposium on Functional and Logic Programming (FLOPS 2010)*, volume 6009 of *LNCS*, pages 134–149, 2010.

[12] Saurin, A. Typing streams in the $\Lambda\mu$-calculus. *ACM Transactions on Computational Logic*, 11:1–34, 2010.

[13] Scott, D.S. Continuous lattices. In *Toposes, Algebraic Geometry, and Logic*, volume 274 of *Lecture Notes in Mathematics*, pages 97–136. 1972.

[14] Smyth, M.B. and Plotkin, G.D. The category-theoretic solution of recursive domain equations. *SIAM Journal on Computation*, 11(4):761–783, 1982.

[15] Streicher, T. and Reus, B. Classical logic, continuation semantics and abstract machines. *Journal of Functional Programming*, 8(6):543–572, November 1998.

[16] van Bakel, S., Barbanera, F., and de'Liguoro, U. A filter model for the $\lambda\mu$-calculus. In C.-H.L. Ong, editor, *Typed Lambda Calculi and Applications, 10th International Conference (TLCA 2011)*, volume 6690 of *LNCS*, pages 213–228, 2011.