Reduction System for Extensional Lambda-mu Calculus

Koji Nakazawa Tomoharu Nagai

Graduate School of Informatics, Kyoto University

Abstract. The $\Lambda\mu$-calculus is an extension of Parigot's $\lambda\mu$-calculus, the untyped variant of which is proved by Saurin to satisfy some fundamental properties such as the standardization and the separation theorem. For the untyped $\Lambda\mu$-calculus, Nakazawa and Katsumata gave extensional models, called stream models, which is a simple extension of the λ-models, and in which terms are represented as functions on streams. However, we cannot construct a term model from the $\Lambda\mu$-calculus. This paper introduces an extension of the $\Lambda\mu$-calculus, called $\Lambda\mu_{\mathsf{cons}}$, on which we can construct a term model as a stream model, and its reduction system satisfying confluence, subject reduction, and strong normalization.

Keywords: Lambda-mu calculus, confluence, strong normalization, extensional models.

1 Introduction

The $\lambda\mu$-calculus was originally introduced by Parigot [16] as a term assignment system for the classical natural deduction, and then a lot of studies have been devoted to the $\lambda\mu$-calculus from both sides of logic and computer science.

An extension of the $\lambda\mu$-calculus was given by de Groote [6] to study continuation-passing-style translations and machines for the calculus. As Saurin showed in [18, 19], an untyped variant of this extension, called the $\Lambda\mu$-calculus, enjoys fundamental properties such as the standardization and the separation theorem. In particular, the latter does not hold for the original $\lambda\mu$-calculus as shown by David and Py in [5].

For the untyped $\Lambda\mu$-calculus, Nakazawa and Katsumata [14] gave a class of extensional models, called *stream models*, in which $\Lambda\mu$-terms are represented as functions on streams. The stream models are simple extension of the λ-models and naturally reflect the idea that the $\Lambda\mu$-terms represent functions on streams. The stream models are similar to the continuation models of [21], and we can see the relationship of them as

$$\frac{\text{continuation models}}{\lambda\mu} = \frac{\text{stream models}}{\Lambda\mu}.$$

Nakazawa and Katsumata showed the soundness and gave an algebraic characterization for the stream models, but a term model cannot be constructed from the $\Lambda\mu$-calculus, and completeness with respect to the stream models has not been discussed.

Regarding types, some type assignment systems for the $\Lambda\mu$-calculus has been introduced. Saurin [20] gave a type system for the $\Lambda\mu$-calculus as a stream calculus, and Gaboardi and Saurin [10] proposed an extension of Saurin's type system with recursive types. De'Liguoro [8] gave an intersection type system and filter models for the $\Lambda\mu$-calculus, based on the stream models. However, the results on the stream models in [14] have not been adapted to typed calculi.

The main results of this paper are the following.

- An extension $\Lambda\mu_{\mathsf{cons}}$ of the $\Lambda\mu$-calculus and its reduction system are proposed. $\Lambda\mu_{\mathsf{cons}}$ induces a term model as a stream model, and hence it is sound and complete with respect to the stream model. The reduction is confluence for the untyped $\Lambda\mu_{\mathsf{cons}}$.
- A type assignment for $\Lambda\mu_{\mathsf{cons}}$ based on de'Liguoro's type system is proposed. The stream models can be adapted to the typed $\Lambda\mu_{\mathsf{cons}}$, and the reduction on the typed $\Lambda\mu_{\mathsf{cons}}$ satisfies subject reduction and strong normalization.

First, we define the equational logic and the reduction system for the untyped $\Lambda\mu_{\mathsf{cons}}$. The calculus $\Lambda\mu_{\mathsf{cons}}$ explicitly contains stream expressions, and this extension is similar to the $\lambda\mu$-calculus of Streicher and Reus [21]. The essential difference from the $\Lambda\mu$-calculus is the surjectivity axiom for streams, which requires any stream S to be expressed as $(\mathsf{car}S) :: (\mathsf{cdr}S)$. In $\Lambda\mu$, bound stream variables satisfy the axiom due to the axiom called (*fst*), but it does not necessarily hold for free stream variables. The reduction system for the untyped $\Lambda\mu_{\mathsf{cons}}$ is confluent for all of terms including open ones in contrast to the $\Lambda\mu$-calculus in [20], where confluence is proved only for the closed terms.

Secondly, a typed variant of $\Lambda\mu_{\mathsf{cons}}$ is proposed. Following de'Liguoro's type system in [8], functional types are from streams to individual data, which matches the idea of the stream models. For types of streams, we adopt (restricted forms of) recursive types for finiteness of the calculus, similarly to Gaboardi and Saurin's type system in [10]. The stream models can be adapted to the typed $\Lambda\mu_{\mathsf{cons}}$, and we prove that the reduction on the typed $\Lambda\mu_{\mathsf{cons}}$ satisfies subject reduction and strong normalization.

2 $\Lambda\mu_{\mathsf{cons}}$

In this section, we will define the untyped calculus $\Lambda\mu_{\mathsf{cons}}$.

The calculus $\Lambda\mu_{\mathsf{cons}}$ can be considered as an extension of Parigot's $\lambda\mu$-calculus, and it is essentially the same as the nil-free fragment of the extended version of the stack calculus in [3]. We use the notation $t\alpha$ to express named a term $[\alpha]t$ in the original $\lambda\mu$-calculus, following the idea from [18] and [4] that it is a function application of t to a stream α.

2.1 Definition of Untyped $\Lambda\mu_{\mathsf{cons}}$

Definition 1 (Untyped $\Lambda\mu_{\mathsf{cons}}$). We are supposed to have two sorts of variables: term variables, denoted by $x, y, z, \cdots$, and stream variables, denoted by

$\alpha, \beta, \cdots$. The set of the term variables and the stream variables are denoted by TV and SV, respectively.

The *terms* and the *streams* of $\Lambda\mu_{\mathsf{cons}}$ are defined as

$$t, u ::= x \mid \lambda x.t \mid ts \mid \mu\alpha.t \mid tS \mid \mathsf{car}S \qquad\qquad S ::= \alpha \mid t :: S \mid \mathsf{cdr}S$$

The set of the $\Lambda\mu_{\mathsf{cons}}$-terms and the set of $\Lambda\mu_{\mathsf{cons}}$-streams are denoted by Tm and St, respectively. Occurrences of x in $\lambda x.t$ and occurrences of α in $\mu\alpha.t$ are considered to be bound. A variable occurrence which is not bound is called *free.* A term which contains no free stream variables is called *stream closed.* The size of t (S) is defined as usual, and it is denoted by $|t|$ ($|S|$, respectively).

The axiom schema of $\Lambda\mu_{\mathsf{cons}}$ are the following:

$$\begin{aligned}
(\lambda x.t)u &= t[x := u] && (\beta_T)\\
(\mu\alpha.t)S &= t[\alpha := S] && (\beta_S)\\
\lambda x.tx &= t && (\eta_T)\\
\mu\alpha.t\alpha &= t && (\eta_S)\\
(\mathsf{car}S) :: (\mathsf{cdr}S) &= S && (\mathsf{surj})\\
\mathsf{cdr}(t :: S) &= S && (\mathsf{cdr})\\
t(u :: S) &= (tu)S && (\mathsf{assoc})
\end{aligned}$$

where, t contains no free x in (η_T), and t contains no free α in (η_S). The reflexive, symmetry, transitive, and compatible relation $=_{\Lambda\mu_{\mathsf{cons}}}$ is defined from the above axiom schema.

We use the following abbreviations

$$\mathsf{cdr}^0 S \equiv S \qquad \mathsf{cdr}^{i+1} S \equiv \mathsf{cdr}(\mathsf{cdr}^i S) \qquad \mathsf{cadr}^i(S) \equiv \mathsf{car}(\mathsf{cdr}^i\, S)$$

for $i \geq 0$. We have $\mathsf{cadr}^i(t_0 :: t_1 :: \cdots :: t_n :: \alpha) =_{\Lambda\mu_{\mathsf{cons}}} t_i$ for $(0 \leq i \leq n)$.

Example 1. 1. In $\Lambda\mu_{\mathsf{cons}}$, the μ-rule $(\mu\alpha.t)u = \mu\alpha.t[\alpha := u :: \alpha]$ is admissible as follows.

$$\begin{aligned}
(\mu\alpha.t)u &=_{\Lambda\mu_{\mathsf{cons}}} \mu\beta.(\mu\alpha.t)u\beta && (\eta_S)\\
&=_{\Lambda\mu_{\mathsf{cons}}} \mu\beta.(\mu\alpha.t)(u :: \beta) && (\mathsf{assoc})\\
&=_{\Lambda\mu_{\mathsf{cons}}} \mu\beta.t[\alpha := u :: \beta] && (\beta_S)\\
&\equiv \mu\alpha.t[\alpha := u :: \alpha].
\end{aligned}$$

2. By the fixed-point combinator Y in the λ-calculus, the n-th function on streams is defined as

$$\mathsf{nth} \equiv Y(\lambda f.\mu\alpha.\lambda n.\mathsf{ifzero}\ n\ \mathsf{then}\ (\mathsf{car}\,\alpha)\ \mathsf{else}\ f(\mathsf{cdr}\,\alpha)(\mathsf{pred}\,n)),$$

where ifzero and pred are supposed to be defined on the Church numerals, and then we have $\mathsf{nth}(t_0 :: \cdots :: t_m :: S)\overline{k} =_{\Lambda\mu_{\mathsf{cons}}} t_k$ for any $k \leq m$, where $\overline{k}$ is the Church numeral representing k.

The calculus $\Lambda\mu_{\mathsf{cons}}$ is a natural extension of the $\Lambda\mu$-calculus, since the $\Lambda\mu$ is the $(::, \mathsf{cdr})$-free fragment of $\Lambda\mu_{\mathsf{cons}}$. The calculus $\Lambda\mu_{\mathsf{cons}}$ is also close to the $\lambda\mu$-calculus treated in [21], which explicitly has the expressions for continuations but no cdr operator. The main difference from these existing calculi is the surjectivity axiom (surj), and hence conservativity of these extensions is not clear. We will discuss the relationship with existing calculi in Section 6.

2.2 Stream Models for Untyped $\Lambda\mu_{\mathsf{cons}}$

The definition of the stream models for the untyped $\Lambda\mu_{\mathsf{cons}}$ is the same as [14] for the untyped $\Lambda\mu$-calculus. We use $\overline{\lambda}$ to denote the meta-level function abstraction.

Definition 2. The set $\mathcal{S}$ is called a *stream set* on a set $\mathcal{D}$ if there is a bijective mapping $(::)$ from $\mathcal{D} \times \mathcal{S}$ to $\mathcal{S}$. For a stream set $\mathcal{S}$, the inverse of $(::)$ is denoted by $\langle \mathsf{Car}, \mathsf{Cdr} \rangle$.

Definition 3. A *stream model* consists of

1. a non-empty set $\mathcal{D}$ and a stream set $\mathcal{S}$ on $\mathcal{D}$,
2. a subset $[\mathcal{S} \to \mathcal{D}]$ of the set of functions from $\mathcal{S}$ to $\mathcal{D}$,
3. $\Phi : \mathcal{D} \to [\mathcal{S} \to \mathcal{D}]$ a bijective mapping, the inverse of which is denoted by Ψ, and we denote $d\star s$ for $\Phi(d)(s)$,

such that the meaning function $[\![\cdot]\!]$ can be defined as follows.

$$\begin{aligned}
[\![x]\!]_\rho &= \rho(x) & [\![\alpha]\!]_\rho &= \rho(\alpha)\\
[\![\lambda x.t]\!]_\rho &= \Psi(\overline{\lambda} s \in \mathcal{S}.[\![t]\!]_{\rho[x \mapsto \mathsf{Car}\, s]}\star(\mathsf{Cdr}\, s)) & [\![t :: S]\!]_\rho &= [\![t]\!]_\rho :: [\![S]\!]_\rho\\
[\![tu]\!]_\rho &= \Psi(\overline{\lambda} s \in \mathcal{S}.[\![t]\!]_\rho\star([\![u]\!]_\rho :: s)) & [\![\mathsf{cdr} S]\!]_\rho &= \mathsf{Cdr}[\![S]\!]_\rho\\
[\![\mu\alpha.t]\!]_\rho &= \Psi(\overline{\lambda} s \in \mathcal{S}.[\![t]\!]_{\rho[\alpha \mapsto s]}) & &\\
[\![tS]\!]_\rho &= [\![t]\!]_\rho\star[\![S]\!]_\rho & &
\end{aligned}$$

The set $\mathsf{Tm}/=_{\Lambda\mu_{\mathsf{cons}}}$ is a stream model, which we call *open term model*, in the case of $\Lambda\mu_{\mathsf{cons}}$ due to the surj axiom.

Proposition 1 (Open term model). *Define the sets*

$$\begin{aligned}
[t] &= \{t' \in \mathsf{Tm} \mid t =_{\Lambda\mu_{\mathsf{cons}}} t'\} & \mathcal{D} &= \{[t] \mid t \in \mathsf{Tm}\}\\
[S] &= \{S' \in \mathsf{St} \mid S =_{\Lambda\mu_{\mathsf{cons}}} S'\} & \mathcal{S} &= \{[S] \mid S \in \mathsf{St}\}\\
[\mathcal{S} \to \mathcal{D}] &= \{f_{[t]} \mid t \in \mathsf{Tm}\}, & &
\end{aligned}$$

where $f_{[t]}$ denotes the function $\overline{\lambda}[S] \in \mathcal{S}.[tS]$, and the functions

$$\begin{aligned}
\Phi([t]) &= f_{[t]} & \mathsf{Car}[S] &= [\mathsf{car} S] & [t] :: [S] &= [t :: S].\\
\Psi(f_{[t]}) &= [t] & \mathsf{Cdr}[S] &= [\mathsf{cdr} S] & &
\end{aligned}$$

This structure $\mathcal{T}$ is a stream model with the meaning function given by $[\![t]\!]_\rho = [t\theta_\rho]$, where θ_ρ is the substitution defined as $\theta_\rho(x) = u$ for $\rho(x) = [u]$ and $\theta_\rho(\alpha) = S$ for $\rho(\alpha) = [S]$. In particular, we have $[\![t]\!] = [t]$ for closed term t.

Proof. Straightforward. Note that the cons operator (::) is bijective, since $\mathsf{Car}[S] :: \mathsf{Cdr}[S] = [(\mathsf{car}S) :: \mathsf{cdr}S] = [S]$ holds by the axiom (surj). Hence, $\mathcal{S}$ is a stream set on $\mathcal{D}$.

Theorem 1 (Soundness and completeness). *For any t and u, $t =_{\Lambda\mu_{\mathsf{cons}}} u$ holds if and only if $[\![t]\!]_\rho = [\![u]\!]_\rho$ holds for any stream model and ρ.*

Proof. ($\Rightarrow$) The soundness is proved by induction on $t = u$. The proof is similar to the case of the $\Lambda\mu$-calculus in [14]. The case of (surj) is proved by that $s = (\mathsf{Car}s) :: (\mathsf{Cdr}s)$ and $[\![\mathsf{car}]\!](d :: s) = d$ hold for any $d \in \mathcal{D}$ and $s \in \mathcal{S}$.

($\Leftarrow$) By Proposition 1, we have $[\![t]\!]^{\mathcal{T}}_\rho = [\![u]\!]^{\mathcal{T}}_\rho$ in the open term model. In particular, if we take ρ as $\rho(x) = [x]$ and $\rho(\alpha) = [\alpha]$, then we have $[t] = [u]$, which means $t =_{\Lambda\mu_{\mathsf{cons}}} u$.

Some properties of the stream models are shown in [14]. One of them guarantees existence of a non-trivial stream model, which gives a semantical proof of the consistency of the equational logic of $\Lambda\mu_{\mathsf{cons}}$.

Proposition 2 ([14]). *For any pointed CPO D, there exists a stream model D^S_∞ into which D can be embedded.*

Corollary 1 (Consistency). *There exist closed $\Lambda\mu_{\mathsf{cons}}$-terms t and u such that $t =_{\Lambda\mu_{\mathsf{cons}}} u$ does not hold.*

Proof. By Proposition 2, there exists a non-trivial stream model containing two different elements d_1 and d_2. If we define ρ as $\rho(x_i) = d_i$ for two distinct variables x_1 and x_2, then we have $[\![x_1]\!]_\rho \neq [\![x_2]\!]_\rho$, so we have $x_1 \neq_{\Lambda\mu_{\mathsf{cons}}} x_2$ by the soundness. Assume that $\lambda xy.x =_{\Lambda\mu_{\mathsf{cons}}} \lambda xy.y$, and then we have $x_1 =_{\Lambda\mu_{\mathsf{cons}}} (\lambda xy.x)x_1x_2 =_{\Lambda\mu_{\mathsf{cons}}} (\lambda xy.y)x_1x_2 =_{\Lambda\mu_{\mathsf{cons}}} x_2$. That contradicts.

3 Reduction System

3.1 Reduction for Untyped $\Lambda\mu_{\mathsf{cons}}$

Definition 4. The one-step reduction $\to$ on terms and streams of $\Lambda\mu_{\mathsf{cons}}$ is the least compatible relation satisfying the following axioms.

$$\begin{aligned}
(\mu\alpha.t)u &\to t[\alpha := u :: \alpha] && (\beta_T)\\
(\mu\alpha.t)S &\to t[\alpha := S] && (\beta_S)\\
\lambda x.t &\to \mu\alpha.t[x := \mathsf{car}\alpha](\mathsf{cdr}\alpha) && (\mathsf{exp})\\
t(u :: S) &\to tuS && (\mathsf{assoc})\\
\mathsf{car}(u :: S) &\to u && (\mathsf{car})\\
\mathsf{cdr}(u :: S) &\to S && (\mathsf{cdr})\\
\mu\alpha.t\alpha &\to t \qquad (\alpha \notin FV(t)) && (\eta_S)\\
(\mathsf{car}S) :: (\mathsf{cdr}S) &\to S && (\eta_{::})\\
t(\mathsf{car}S)(\mathsf{cdr}S) &\to tS && (\eta'_{::})
\end{aligned}$$

Here, α in (exp) is a fresh stream variable. The relation $\to^*$ is the reflexive transitive closure of $\to$, the relation $\to^+$ is the transitive closure of $\to$, and the relation $\to^=$ is the reflexive closure of $\to$.

2. The relations $\to_\beta$ and $\to_\eta$ are defined as the least compatible relations satisfying the following axioms, respectively.
$\to_\beta$: (β_T), (β_S), (exp), (assoc), (car), and (cdr)
$\to_\eta$: (η_S), $(\eta_{::})$, $(\eta'_{::})$, (car), and (cdr)
We also use $\to_\beta^*$, $\to_\eta^+$, and so on.

Note that $\to_\beta$-normal forms are characterized by

$$t ::= a \mid \mu\alpha.t \qquad\qquad a ::= x \mid \mathsf{cadr}^n\alpha \mid at \mid a(\mathsf{cdr}^n\alpha).$$

It is known that adding naïvely the η-rule $\lambda x.tx \to t$ to the $\lambda\mu$-calculus destroys confluence [5]. The counterexample is

$$\begin{tikzcd} \lambda x.(\mu\alpha.y\beta)x \arrow[r, "\eta"] \arrow[d, "\beta"] & \mu\alpha.y\beta \ , \\ \lambda x.\mu\alpha.y\beta & \end{tikzcd}$$

and an expansion rule called (*fst*) in [20] is proposed to recover the confluence. The rule (*fst*), which is expressed in $\Lambda\mu_{\mathsf{cons}}$ as

$$\mu\alpha.t \to \lambda x.\mu\alpha.t[\alpha := x :: \alpha] \qquad\qquad (fst),$$

seems natural since it means the surjectivity of the bound variable α. However, if we consider type systems, it induces a type dependent reduction for subject reduction and strong normalization. Alternatively, we adopt (exp) with a help of the explicit syntax of streams in $\Lambda\mu_{\mathsf{cons}}$. The above critical pair is solved as

$$\begin{aligned} \lambda x.\mu\alpha.y\beta &\to \mu\gamma.(\mu\alpha.y\beta)(\mathsf{cdr}\gamma) && (\mathsf{exp}) \\ &\to \mu\gamma.y\beta (= \mu\alpha.y\beta) && (\beta_S). \end{aligned}$$

We can easily see that the following are derivable:

$$(\lambda x.t)u \to^* t[x := u] \qquad\qquad \lambda x.tx \to^* t \qquad (x \notin FV(t)),$$

and hence, the following holds.

Proposition 3. *The equivalence closure of $\to$ coincides to the extensional equality of $\Lambda\mu_{\mathsf{cons}}$.*

3.2 Confluence

In this subsection, we prove confluence of the reduction $\to$. In contrast to the $\Lambda\mu$-calculus [20], the result is not restricted to terms without free stream variables. The proof is not straightforward because of the non left-linear axioms $(\eta_{::})$.

Proposition 4. *$\to_\beta$ and $\to_\eta$ are respectively confluent.*

Proof. (β) By a generalized notion of complete development, which is independently introduced in [7] and [11]. We can define the complete development $(\cdot)^\dagger$ for the reduction $\to_\beta$, and show that $t \to_\beta u$ implies $u \to_\beta^* t^\dagger \to_\beta^* u^\dagger$, from which the confluence of $\to_\beta$ follows. Only non-trivial point in the definition of $(\cdot)^\dagger$ is that we exceptionally define $((\mu\alpha.t)uS)^\dagger = t^\dagger[\alpha := u^\dagger :: S^\dagger]$ (not $(\mu\alpha.t^\dagger[\alpha := u^\dagger :: \alpha])S^\dagger$), since we have to show that $((\mu\alpha.t)(u :: S))^\dagger \to^* ((\mu\alpha.t)uS)^\dagger$, the right-hand side of which is $t^\dagger[\alpha := (u :: S)^\dagger] = t^\dagger[\alpha := u^\dagger :: S^\dagger]$.

(η) Since $\to_\eta$ is strongly normalizing, it is sufficient to prove weak confluence. It is straightforward.

In order to prove commutativity of β and η, we define some auxiliary notions.

Definition 5. 1. The relation $\to_{\eta^-}$ is the least compatible relation satisfying (η_S), (car), (cdr), and the restricted forms of ($\eta_{::}$) and ($\eta'_{::}$) as follows.

$$(\mathsf{car}S) :: (\mathsf{cdr}S) \to S \qquad (S = \mathsf{cdr}^n\alpha) \qquad (\eta_{::}^-)$$
$$t(\mathsf{car}S)(\mathsf{cdr}S) \to tS \qquad (S = \mathsf{cdr}^n\alpha) \qquad (\eta_{::}'^-)$$

2. The translation S^c on streams is defined as follows.

$$\alpha^c = \alpha \qquad (t :: S)^c = t :: S \qquad (\mathsf{cdr}S)^c = \begin{cases} S'^c & S^c = t' :: S' \\ \mathsf{cdr}^{n+1}\alpha & S^c = \mathsf{cdr}^n\alpha \end{cases}$$

3. The relation $\to_c$ is the least compatible relation satisfying

$$S \to S^c \qquad (S \neq S^c) \qquad (\mathsf{c}).$$

Lemma 1. *For any S, we have the following.*
1. *S^c is well-defined.*
2. *$|S^c| \le |S|$.*
3. *S^c is of the form either $t' :: S'$ or $\mathsf{cdr}^n\alpha$ ($n \ge 0$).*
4. *$S \to^* S^c$ holds by (cdr), and hence we have $S \to_{\eta^-}^* S^c$ and $S \to_\beta^* S^c$.*
5. *$(\mathsf{car}S^c) :: (\mathsf{cdr}S^c) \to_{\eta^-}^* S^c$ holds.*
6. *$S \to_{\eta^-} S'$ implies $S^c \to_{\eta^-}^= S'^c$.*

Proof. 1, 2, and 3 are simultaneously proved by induction on $|S|$. 4 is also by induction on $|S|$. 5 immediately follows from 3.

6 is proved by induction on $|S|$. The only complicated case is $\mathsf{cdr}S \to_{\eta^-} \mathsf{cdr}S'$ where $S \to_{\eta^-} S'$. By the induction hypothesis for S, we have $S^c \to_{\eta^-}^= S'^c$.

If $S^c = t_0 :: S_0$, we have two cases in the form of $S^c \to_{\eta^-}^= S'^c$: (i) $S'^c = t'_0 :: S'_0$ where either $t_0 \to_{\eta^-}^= t'_0$ or $S_0 \to_{\eta^-}^= S'_0$, and (ii) $S^c = \mathsf{car}\,\mathsf{cdr}^n\alpha :: \mathsf{cdr}\,\mathsf{cdr}^n\alpha$ and $S'^c = \mathsf{cdr}^n\alpha$. In the case of (i), we have $(\mathsf{cdr}S')^c = S_0'^c$, so $(\mathsf{cdr}S)^c \to_{\eta^-}^= (\mathsf{cdr}S')^c$ holds by the induction hypothesis for S_0, the size of which is strictly smaller than $|\mathsf{cdr}S|$. In the case of (ii), we have $(\mathsf{cdr}S)^c = \mathsf{cdr}^{n+1}\alpha = (\mathsf{cdr}S')^c$.

If $S^c = \mathsf{cdr}^n\alpha$, S'' is also $\mathsf{cdr}^n\alpha$, so we have $S' \to_c^= \mathsf{cdr}^n\alpha$. Then, we have $(\mathsf{cdr}S)^c = \mathsf{cdr}^{n+1}\alpha$ and $\mathsf{cdr}S' \to_c^= \mathsf{cdr}^{n+1}\alpha$.

Note that $t \to_{\eta^-} t'$ does not necessarily imply $t[\alpha := S] \to_{\eta^-} t'[\alpha := S]$ since we restrict the $\eta_{::}$-reduction. Instead, we have the following lemma.

Lemma 2. *If $t \to_{\eta^-} t'$, then $t[\alpha := S] \to_{\eta^-} u$ and $t'[\alpha := S] \to_c^* u$ for some u.*

Proof. By induction on $t \to_{\eta^-} t'$. For the case of $\eta_{::}^-$, suppose $t = \mathsf{car}\,\mathsf{cdr}^n\alpha :: \mathsf{cdr}\,\mathsf{cdr}^n\alpha$ and $t' = \mathsf{cdr}^n\alpha$. By 4 and 5 of Lemma 1, we have $t[\alpha := S] \to_{\eta^-}^* \mathsf{car}(\mathsf{cdr}^n S)^c :: \mathsf{cdr}(\mathsf{cdr}^n S)^c \to_{\eta^-}^* (\mathsf{cdr}^n S)^c$ and $t'[\alpha := S] \to_c (\mathsf{cdr}^n S)^c$. The case of $\eta'^-_{::}$ is similarly proved, and the other cases are straightforward.

Lemma 3. *The following commuting diagrams hold.*

1.
$$\begin{tikzcd}
\cdot \arrow[r, "c"] \arrow[d, "\eta^-"'] & \cdot \arrow[d, dotted, "="', "\eta^-"] \\
\cdot \arrow[r, dotted, "="', "c"] & \exists\cdot
\end{tikzcd}$$

2.
$$\begin{tikzcd}
\cdot \arrow[rr, "\beta"] \arrow[d, "\eta^-"'] & & \cdot \arrow[d, dotted, "*"', "\eta^-"] \\
\cdot \arrow[r, dotted, "="', "\beta"] & \cdot \arrow[r, dotted, "*"', "c"] & \exists\cdot
\end{tikzcd}$$

3.
$$\begin{tikzcd}
\cdot \arrow[r, "c", "*"'] \arrow[d, "\eta"', "*"] & \cdot \arrow[d, dotted, "*"', "\eta^-"] \\
\cdot \arrow[r, dotted, "*", "c"'] & \exists\cdot
\end{tikzcd}$$

Proof. 1. By induction on the size of terms and streams at the start point, using Lemma 1.6.

2. By induction on the terms and streams at the start point. We only show the case of $(\mu\alpha.t)S \to_\beta t[\alpha := S]$ and $(\mu\alpha.t)S \to_{\eta^-} (\mu\alpha.t')S$ where $t \to_{\eta^-} t'$. By Lemma 2, there exists u such that $t[\alpha := S] \to_{\eta^-}^* u$ and $(\mu\alpha.t')S \to_\beta t'[\alpha := S] \to_c^* u$.

3. By 1, $\to_{\eta^-}^*$ and $\to_c^*$ commute, so it is sufficient to consider each step of $(\eta_{::})$ and $(\eta'_{::})$ which is not restricted. It is proved by the fact that $t \to_\eta t'$ implies that there exists u such that $t \to_{\eta^-}^* u$ and $t' \to_c u$. It is proved straightforwardly.

The point is that, if we consider the full η-reduction, $\to^=$ in 1 and 2 does not hold, but only $\to^*$ holds.

Proposition 5. *$\to_\beta^*$ and $\to_\eta^*$ commute.*

Proof. By 1 and 2 of Lemma 3, $\to_\beta^*$ and $\to_{\eta^-}^*$ commute. By this and Lemma 3.3, $\to_\beta^*$ and $\to_\eta^*$ commute as follows.

$$\begin{tikzcd}
\cdot \arrow[rr, "\beta"] \arrow[d, "\eta"', "*"] \arrow[dr, dotted, "\eta^-\,*"] & & \cdot \arrow[d, dotted, "*"', "\eta^-"] \\
\cdot \arrow[r, dotted, "*", "c"'] & \cdot \arrow[r, dotted, "*", "\beta"'] & \cdot
\end{tikzcd}$$

Theorem 2 (Confluence). *The reduction $\to$ is confluent.*

Proof. It follows from Proposition 4 and 5.

It gives syntactic proof of consistency of the equational logic of $\Lambda\mu_{\mathsf{cons}}$.

Corollary 2 (Consistency of $\Lambda\mu_{\mathsf{cons}}$). *There exists two closed $\Lambda\mu_{\mathsf{cons}}$-terms t and u such that $t =_{\Lambda\mu_{\mathsf{cons}}} u$ does not hold.*

Proof. $\mu\alpha.\mathsf{car}\alpha$ and $\mu\alpha.\mathsf{cadr}\alpha$ are different normal forms, and hence they are not related by $=_{\Lambda\mu_{\mathsf{cons}}}$.

4 Typed $\Lambda\mu_{\mathsf{cons}}$

We will give a type assignment system for $\Lambda\mu_{\mathsf{cons}}$, which is inspired by the type system for the $\Lambda\mu$-calculus of de'Liguoro in [8], and adopting recursive types to represent types of streams like [10]. The reduction introduced in the previous section enjoys subject reduction and confluence with respect to the type system.

4.1 Definition of Typed $\Lambda\mu_{\mathsf{cons}}$

The types of streams will be introduced as non-empty lists of types of individual data such as $[\delta_0, \delta_1]$, which is a special case of the recursive types, and which is just an abbreviation for $\mu\chi.\delta_0 \times \delta_1 \times \chi$. The following axiomatization for the equivalence on the stream types are based on the idea borrowed from well-known results for recursively defined trees in [17, 12, 1], and [2].

Definition 6 (Typed $\Lambda\mu_{\mathsf{cons}}$). The types consist of two sorts, *term types* and *stream types*, which are inductively defined as

$$\delta ::= X \mid \sigma \to \delta \qquad\qquad \sigma ::= [\delta_0, \cdots, \delta_{n-1}] \mid \delta \times \sigma,$$

where $[\delta_0, \cdots, \delta_{n-1}]$ is a non-empty finite list of types. The relation $\sim$ on the types and the stream types is defined as the smallest compatible equivalence relation satisfying the following rule schemata.

$$\frac{}{[\delta_0, \cdots, \delta_{n-1}] \sim \delta_0 \times [\delta_1, \cdots \delta_{n-1}, \delta_0]}\ (\text{Fld}) \qquad \frac{\delta_0 \times \cdots \delta_{n-1} \times \sigma \sim \sigma}{[\delta_0, \cdots, \delta_{n-1}] \sim \sigma}\ (\text{Ctr})$$

A term context Γ and a stream context Δ are finite lists of pairs of the form $(x : \delta)$ and $(\alpha : \sigma)$, respectively, in which each variable occurs at most once.

The typing rules of $\Lambda\mu_{\mathsf{cons}}$ are the following.

$$\frac{}{\Gamma, x : \delta; \Delta \vdash x : \delta}\ (\text{Ax}) \qquad \frac{}{\Gamma; \Delta, \alpha : \sigma \vdash \alpha : \sigma}\ (\text{SAx})$$

$$\frac{\Gamma, x : \delta; \Delta \vdash t : \sigma \to \delta'}{\Gamma; \Delta \vdash \lambda x.t : \delta \times \sigma \to \delta'}\ (\text{Lam}) \qquad \frac{\Gamma; \Delta \vdash t : \delta \times \sigma \to \delta' \quad \Gamma; \Delta \vdash u : \delta}{\Gamma; \Delta \vdash tu : \sigma \to \delta'}\ (\text{App})$$

$$\frac{\Gamma; \Delta, \alpha : \sigma \vdash t : \delta}{\Gamma; \Delta \vdash \mu\alpha.t : \sigma \to \delta}\ (\text{Mu}) \qquad \frac{\Gamma, \Delta \vdash t : \sigma \to \delta \quad \Gamma, \Delta \vdash S : \sigma}{\Gamma; \Delta \vdash tS : \delta}\ (\text{SApp})$$

$$\frac{\Gamma; \Delta \vdash t : \delta \quad \Gamma; \Delta \vdash S : \sigma}{\Gamma; \Delta \vdash t :: S : \delta \times \sigma}\ (\text{Cons}) \qquad \frac{\Gamma; \Delta \vdash S : \delta \times \sigma}{\Gamma; \Delta \vdash \mathsf{cdr} S : \sigma}\ (\text{Cdr})$$

$$\frac{\Gamma; \Delta \vdash t : \delta \quad \delta \sim \delta'}{\Gamma; \Delta \vdash t : \delta'}\ (\text{TEq}) \qquad \frac{\Gamma; \Delta \vdash S : \sigma \quad \sigma \sim \sigma'}{\Gamma; \Delta \vdash S : \sigma'}\ (\text{SEq})$$

The relation $\Gamma; \Delta \vdash t_1 = t_2 : \delta$ means that $\Gamma; \Delta \vdash t_i : \delta$ $(i = 1, 2)$ and $t_1 =_{\Lambda\mu_{\mathsf{cons}}} t_2$.

The choice of the equivalence is not essential for the following discussion, and we can adopt the equivalence defined by only the fold/unfold axiom as [10]. In fact, the properties shown in this paper can be proved for more general setting in which types of streams are represented as infinite product types, called the expanded types as follows.

Definition 7. *1. The expanded types are defined by*

$$\delta ::= X \mid \sigma \to \delta \qquad\qquad \sigma ::= \mathit{\Pi}_{i\in\mathbb{N}}\delta_i.$$

We also use the notation like $\delta_0 \times \cdots \times \delta_{n-1} \times \mathit{\Pi}_{i\in\mathbb{N}}\delta'_i$.

2. Given a stream type σ *and* $i \in \mathbb{N}$, *we define* $(\sigma)_i$ *by*

$$([\delta_0, \cdots, \delta_{n-1}])_i = \delta_{i \bmod n} \qquad (\delta \times \sigma)_i = \begin{cases} \delta & (i = 0) \\ (\sigma)_{i-1} & (i > 0), \end{cases}$$

where $i \bmod n$ *denotes the remainder of the division of* i *by* n. *We also define the function* $(\sigma)_i$ *for the expanded types as* $(\mathit{\Pi}_{j\in\mathbb{N}}\delta_j)_i = \delta_i$.

3. The expansion of the types is defined as follows.

$$\langle\!|X|\!\rangle = X \qquad \langle\!|\sigma|\!\rangle = \mathit{\Pi}_{i\in\mathbb{N}}\,\langle\!|(\sigma)_i|\!\rangle \qquad \langle\!|\sigma \to \delta|\!\rangle = \langle\!|\sigma|\!\rangle \to \langle\!|\delta|\!\rangle$$

Note that the relation $\sqsubset$ on expanded types defined as $\sigma, \delta \sqsubset \sigma \to \delta$ and $\delta_i \sqsubset \mathit{\Pi}_{i\in\mathbb{N}}\delta_i$ is a well-founded order, and we will use the induction on this order.

Proposition 6. $\delta \sim \delta'$ *(*$\sigma \sim \sigma'$*) iff* $\langle\!|\delta|\!\rangle = \langle\!|\delta'|\!\rangle$ *(*$\langle\!|\sigma|\!\rangle = \langle\!|\sigma'|\!\rangle$*, repectively).*

It is a corollary of the completeness of the axiomatization, for example, in [1], and we can also prove it directly. (See Appendix A for more details.)

Example 2. In [14], SCL is proposed as a combinatory calculus which is equivalent to the $\Lambda\mu$-calculus. However, some constants of SCL are not typable and some of the others have only restricted types in the original typed $\lambda\mu$-calculus. On the other hand, the $\Lambda\mu_{\mathsf{cons}}$-terms corresponding to SCL-constants are typable such as

$$\begin{array}{ll} (\mathsf{K}_1) & ;\vdash \lambda x.\mu\alpha.x : \delta \times \sigma \to \delta \\ (\mathsf{W}_1) & ;\vdash \lambda x.\mu\alpha.x\alpha\alpha : (\sigma \to \sigma \to \delta) \times \sigma \to \delta \end{array}$$

for any term types δ and any stream types σ.

We discuss the relationship with the existing type systems in Section 6 and in [13].

4.2 Stream models for Typed $\Lambda\mu_{\mathsf{cons}}$

The stream models are adapted to the typed $\Lambda\mu_{\mathsf{cons}}$. We only give a brief introduction, and more details are in [13].

A stream model for the typed $\Lambda\mu_{\mathsf{cons}}$ consists of

– families of sets $\{\mathcal{A}^\delta\}_\delta$ and $\{\mathcal{A}^\sigma\}_\sigma$ indexed by the expanded types
– an operation $(\star) : \mathcal{A}^{\sigma\to\delta} \times \mathcal{A}^\sigma \to \mathcal{A}^\delta$ for each σ and δ such that

$$\forall f, g \in \mathcal{A}^{\sigma\to\delta}.[\forall s \in \mathcal{A}^\sigma.[f\star s = g\star s] \Rightarrow f = g].$$

– a bijection $(::) : \mathcal{A}^\delta \times \mathcal{A}^\sigma \to \mathcal{A}^{\delta\times\sigma}$ for each δ and σ, the inverse of which consists of the projection functions $\langle \mathsf{Car}, \mathsf{Cdr}\rangle$.
– a meaning function $[\![\cdot]\!]$ such that $[\![\lambda x^{\delta'}.t^{\sigma\to\delta}]\!]_\rho \in \mathcal{A}^{\delta'\times\sigma\to\delta}$ and $[\![\lambda x^{\delta'}.t^{\sigma\to\delta}]\!]_\rho\star s = [\![t]\!]_{\rho[x\mapsto\mathsf{Car}(s)]}\star\mathsf{Cdr}(s)$ for any $s \in \mathcal{A}^{\delta'\times\sigma}$, and so on.

In particular, a stream model is called *full* if $\mathcal{A}^{\sigma\to\delta}$ is the whole function space from $\mathcal{A}^\sigma$ to $\mathcal{A}^\delta$ for any σ and δ, and $\mathcal{A}^\sigma$ is $\Pi_{i\in\mathbb{N}}\mathcal{A}^{(\sigma)_i}$ for any σ.

Then, in the same way as the untyped case, we can construct the open term model, and prove soundness and completeness as Theorem 1.

Furthermore, we can show the following, corresponding to the well-known Friedman's theorem [9]: the extensional equality in $\lambda^\to$ is characterized by an arbitrary individual full type hierarchy with infinite bases. This theorem is proved by giving the logical relation on the stream models between the open term model and the full stream model.

Theorem 3 (Friedman's theorem for $\Lambda\mu_{\mathsf{cons}}$). *Suppose that a stream model $\mathcal{F}$ is full and all of $\mathcal{F}^X$ are infinite. Then, for any closed typable t and u, $t =_{\Lambda\mu_{\mathsf{cons}}} u$ holds if and only if $[\![t]\!]^\mathcal{F} = [\![u]\!]^\mathcal{F}$ holds.*

5 Reduction for Typed $\Lambda\mu_{\mathsf{cons}}$

In this section, we show two fundamental properties of the reduction on the typed $\Lambda\mu_{\mathsf{cons}}$, subject reduction and strong normalization.

5.1 Subject Reduction

We omit the proof of the subject reduction since it is straightforwardly proved using the usual generation lemma. (See Appendix B for more details.)

Theorem 4 (Subject reduction). *If $\Gamma \mid \Delta \vdash t : \delta$ and $t \to u$ hold, then we have $\Gamma \mid \Delta \vdash u : \delta$.*

5.2 Strong Normalization

First, we prove the strong normalization of $\to_\beta$ by the usual reducibility, and then extend to the full reduction $\to$. The set of terms and streams which are strongly normalizable with respect to $\to_\beta$ are denoted by $\mathsf{SN}^\beta_\mathsf{T}$ and $\mathsf{SN}^\beta_\mathsf{S}$, respectively. Moreover, $\mathsf{SN}^\beta_\mathsf{C}$ is the set of the strongly normalizable stream contexts defined by $C ::= [] \mid CS$ where $S \in \mathsf{SN}^\beta_\mathsf{S}$.

Definition 8. *The relation* Red *indexed by the expanded types are defined as follows:*

$\mathsf{Red}_X = \mathsf{SN}^\beta_\mathsf{T}$,

$t \in \mathsf{Red}_{\sigma\to\delta}$ *iff, for any* $S \in \mathsf{Red}_\sigma$, $tS \in \mathsf{Red}_\delta$,

$S \in \mathsf{Red}_\sigma$ *iff, for any* $n \geq 0$, $\mathsf{cadr}^n S \in \mathsf{Red}_{(\sigma)_n}$.

By the definition, it is easy to see that $S \in \mathsf{Red}_{\delta\times\sigma}$ iff $\mathsf{car}S \in \mathsf{Red}_\delta$ and $\mathsf{cdr}S \in \mathsf{Red}_\sigma$.

Lemma 4. *1.* $\mathsf{Red}_\delta \subseteq \mathsf{SN}^\beta_\mathsf{T}$ *and* $\mathsf{Red}_\sigma \subseteq \mathsf{SN}^\beta_\mathsf{S}$ *hold.*

2. For any $C \in \mathsf{SN}^\beta_\mathsf{C}$, $C[x] \in \mathsf{Red}_\delta$ *and* $C[\mathsf{cadr}^n\alpha] \in \mathsf{Red}_\delta$ *hold.*

3. $\alpha \in \mathsf{Red}_\sigma$ *holds.*

Proof. The proof is given in Appendix C.

Lemma 5. *For any expanded types* δ, σ, *and any* $C \in \mathsf{SN}^\beta_\mathsf{C}$, *the following hold.*

1. For any $u \in \mathsf{SN}^\beta_\mathsf{T}$, $C[\mu\alpha.t[\alpha := u :: \alpha]] \in \mathsf{Red}_\delta$ *implies* $C[(\mu\alpha.t)u] \in \mathsf{Red}_\delta$.

2. For any $S \in \mathsf{SN}^\beta_\mathsf{S}$, $C[t[\alpha := S]] \in \mathsf{Red}_\delta$ *implies* $C[(\mu\alpha.t)S] \in \mathsf{Red}_\delta$.

3. $C[\mu\alpha.t[x := \mathsf{car}\alpha](\mathsf{cdr}\alpha)] \in \mathsf{Red}_\delta$ *implies* $C[\lambda x.t] \in \mathsf{Red}_\delta$.

4. For any $S \in \mathsf{SN}^\beta_\mathsf{S}$, *if* $C[t] \in \mathsf{Red}_\delta$ *implies* $C[\mathsf{car}(t :: S)] \in \mathsf{Red}_\delta$.

5. For any $t \in \mathsf{SN}^\beta_\mathsf{T}$, *if* $C[\mathsf{cadr}^n S] \in \mathsf{Red}_\delta$ *implies* $C[\mathsf{cadr}^{n+1}(t :: S)] \in \mathsf{Red}_\delta$.

Proof. We give only the proof of 2, and the others are proved similarly. In this proof, $\#t$ for $t \in \mathsf{SN}^\beta_\mathsf{T}$ denotes the maximum length of reduction sequences from t, and $\#S$ for $S \in \mathsf{SN}^\beta_\mathsf{S}$ is similarly defined.

First, we show that, for any $S \in \mathsf{SN}^\beta_\mathsf{S}$, $C[t[\alpha := S]] \in \mathsf{SN}^\beta_\mathsf{T}$ implies $C[(\mu\alpha.t)S] \in \mathsf{SN}^\beta_\mathsf{T}$, by induction on the triple of $\#S$, $|S|$, and $\#C[t[\alpha := S]]$ equipped with the lexicographical order. It is sufficient to show that $u \in \mathsf{SN}^\beta_\mathsf{T}$ for any u such that $C[(\mu\alpha.t)S] \to_\beta u$.

Case $C[(\mu\alpha.t)S] \to_\beta C[t[\alpha := S]]$. $C[t[\alpha := S]] \in \mathsf{SN}^\beta_\mathsf{T}$ directly follows from the assumption.

Case $C[(\mu\alpha.t)S] \to_\beta C'[(\mu\alpha.t')S]$. We have $C[t[\alpha := S]\overline{S}] \to_\beta C'[t'[\alpha := S]]$, and hence $C'[(\mu\alpha.t')S] \in \mathsf{SN}^\beta_\mathsf{T}$ follows from the induction hypothesis since $\#C[t[\alpha := S]] > \#C'[t'[\alpha := S]]$.

Case $C[(\mu\alpha.t)S] \to_\beta C[(\mu\alpha.t)S']$. It follows from the induction hypothesis since $\#S > \#S'$.

Case $C[(\mu\alpha.t)(t_0 :: S_0)] \to_\beta C[(\mu\alpha.t)t_0S_0]$ by (assoc). Since $\#(t_0 :: S_0) \geq \#S_0$ and $|t_0 :: S_0| > |S_0|$, we have $C[(\mu\alpha.t[\alpha := t_0 :: \alpha])S_0] \in \mathsf{SN}^\beta_\mathsf{T}$ by the induction hypothesis. Since $t_0 \in \mathsf{SN}^\beta_\mathsf{T}$, we have $C[(\mu\alpha.t)t_0S_0] \in \mathsf{SN}^\beta_\mathsf{T}$ by 1.

Secondly, the lemma is proved by induction on δ. The base case is shown above, and the induction step is straightforward.

Definition 9. $\mathsf{Red}_{\Gamma|\Delta}$ *denotes the set of substitutions* θ *such that* $\theta(x) \in \mathsf{Red}_\delta$ *for any* $x : \delta \in \Gamma$ *and* $\theta(\alpha) \in \mathsf{Red}_\sigma$ *for any* $\alpha : \sigma \in \Delta$.

Proposition 7. *If* $\Gamma \mid \Delta \vdash t : \delta$ *and* $\theta \in \mathsf{Red}_{\Gamma|\Delta}$, *then we have* $t\theta \in \mathsf{Red}_\delta$.

Proof. By induction on the derivation of $\Gamma \mid \Delta \vdash t : \delta$, using Lemma 5.

Theorem 5. *Every typable term t is in $\mathsf{SN}^{\beta}_{\mathsf{T}}$.*

Proof. By Lemma 4, the identity substitution θ is in $\mathsf{Red}_{\Gamma|\Delta}$ for any Γ and Δ. Hence, by Proposition 7, we have $t = t\theta \in \mathsf{Red}_{\delta} \subseteq \mathsf{SN}^{\beta}_{\mathsf{T}}$.

Theorem 6 (Strong normalization). *Every typable term t is strongly normalizing with respect to $\to$.*

Proof. For any reduction sequence, we can postpone any η-reduction, that is, we can prove that $t \to_{\eta} \cdot \to_{\beta} u$ implies $t \to^{+}_{\beta} \cdot \to^{*}_{\eta} u$. Since $\to_{\eta}$ is strongly normalizable, if we have an infinite sequence of $\to$, we can construct an infinite sequence of $\to_{\beta}$, that contradicts Theorem 5.

6 Relationship with Existing Systems

In this section, we discuss the relationship between $\Lambda\mu_{\mathsf{cons}}$ and the existing related systems such as the stack calculus in [4, 3], the untyped $\Lambda\mu$-calculus in [18]. Parigot's original typed $\lambda\mu$-calculus [16], and Gaboardi and Saurin's $\Lambda_{\mathcal{S}}$ in [10].

6.1 Extended Stack Calculus

The calculus $\Lambda\mu_{\mathsf{cons}}$ is essentially an extension of the nil-free fragment of the extended stack calculus in [3]. The stack calculus contains neither term variables, λ-abstractions, nor term applications, but they can be simulated. It is straightforward to see that the reduction of $\Lambda\mu_{\mathsf{cons}}$ is conservative over the stack calculus, that is, for terms t and u of the extended stack calculus without nil, $t \to^{*} u$ in the stack calculus if and only if $t \to^{*} u$ in $\Lambda\mu_{\mathsf{cons}}$. Moreover, our type system can be adapted to the extended stack calculus without nil in a straightforward way. The discussion in this paper on the stream models for the untyped and typed variants of $\Lambda\mu_{\mathsf{cons}}$ can be adapted to the extended stack calculus.

6.2 Untyped $\Lambda\mu$-Calculus

Let $K_0 = \lambda xy.x$ and $K_1 = \lambda x.\mu\alpha.x$. The following is an equation in $\Lambda\mu$ corresponding to a part of the axiom (surj).

$$x\alpha = K_0(x(K_1\alpha))\alpha \qquad (\mathsf{surj}')$$

The equation (surj') does not seem to hold in $\Lambda\mu$, whereas it holds in $\Lambda\mu_{\mathsf{cons}}$, since $K_1\alpha =_{\Lambda\mu_{\mathsf{cons}}} \mathsf{car}\alpha$. However, if we consider only stream closed terms, (surj') is derivable in $\Lambda\mu$ as

$$\begin{aligned}
&\mu\alpha.\cdots K_0(x(K_1\alpha))\alpha\cdots \\
=\,&\lambda y.\mu\alpha.\cdots K_0(x(K_1\, y\alpha))y\alpha\cdots \\
=\,&\lambda y.\mu\alpha.\cdots xy\alpha\cdots \\
=\,&\mu\alpha.\cdots x\alpha\cdots\,.
\end{aligned}$$

We can prove that $\Lambda\mu_{\mathsf{cons}}$ is a conservative extension over the $\Lambda\mu$-calculus for the stream closed terms. (In Appendix D, we will give the proof.)

Proposition 8 (Conservativity over $\Lambda\mu$). *For any stream closed $\Lambda\mu$-terms t and u, $t = u$ holds in $\Lambda\mu_{\mathsf{cons}}$ if and only if $t = u$ holds in $\Lambda\mu$.*

Corollary 3. *For any stream closed $\Lambda\mu$-terms t and u, $t = u$ holds in the $\Lambda\mu$-calculus if and only if $[\![t]\!]_\rho = [\![u]\!]_\rho$ for any stream model $\mathcal{A}$ and ρ.*

Saurin [18] proved the separation theorem of the $\Lambda\mu$-calculus. By the conservativity, $\Lambda\mu_{\mathsf{cons}}$ inherits the separation theorem from $\Lambda\mu$ for stream closed terms. The *canonical normal forms* in the $\Lambda\mu$-calculus are defined as terms which are η-normal and contain no pre-redexes in the sense of [19], that is, subterms in the form of either $(\lambda x.t)u$, $(\lambda x.t)\beta$, $(\mu\alpha.t)u$, or $(\mu\alpha.t)\beta$.

Theorem 7 (Separation theorem for $\Lambda\mu$, [20]). *Let $\Lambda\mu$-terms t_1 and t_2 be stream closed canonical normal forms. If $t_1 = t_2$ does not hold in $\Lambda\mu$, then there exists an applicative context C such that $C[t_1] \to^* \lambda xy.x$ and $C[t_2] \to^* \lambda xy.y$ hold in $\Lambda\mu$, where the applicative contexts are defined as $C ::= [] \mid Ct \mid C\alpha$.*

By this theorem, the separation theorem for $\Lambda\mu_{\mathsf{cons}}$ is proved. (For detailed proof, see Appendix D.)

Theorem 8 (Separation theorem for $\Lambda\mu_{\mathsf{cons}}$). *Let t_1 and t_2 be distinct stream closed normal $\Lambda\mu_{\mathsf{cons}}$-terms. For any normal $\Lambda\mu_{\mathsf{cons}}$-terms u_1 and u_2, there exists an applicative context C such that $C[t_1] \to^* u_1$ and $C[t_2] \to^* u_2$ hold in $\Lambda\mu_{\mathsf{cons}}$.*

6.3 Type Assignment for $\Lambda\mu$-Calculus

On the typed calculi, we can find more detailed discussion in [13].

In our typed $\Lambda\mu_{\mathsf{cons}}$, only functional types from streams to individual data are considered, inspired by the type system of de'Liguoro [8]. However, every typable term in Pagani and Saurin's $\Lambda_{\mathcal{S}}$ [15, 20] is also typable in $\Lambda\mu_{\mathsf{cons}}$. Therefore, every typable term in the first-order fragment of Parigot's $\lambda\mu$-calculus [16] is also typable in $\Lambda\mu_{\mathsf{cons}}$.

Here, we show the translation from the $\lambda\mu$-calculus to $\Lambda\mu_{\mathsf{cons}}$, which is based on the same idea of translations in Saurin [20] and van Bakel et al. [23]. They show that their type systems correspond to the image of negative translations from the classical logic to the intuitionistic logic. The following negative translation corresponds to the continuation-passing-style translation due to Thielecke [22].

Definition 10 (Negative translation). *We fix a type variable O and an arbitrary stream type θ, and we write $\neg\sigma$ for $\sigma \to O$. The negative translations $\overline{(\cdot)}$ from the implicational formulas to term types in $\Lambda\mu_{\mathsf{cons}}$ are defined as*

$$\overline{A} = \neg A^\bullet \qquad p^\bullet = \theta \qquad (A \to B)^\bullet = \neg A^\bullet \times B^\bullet.$$

Proposition 9. *If $\Gamma \vdash t : A; \Delta$ holds in the Parigot's original typed $\lambda\mu$-calculus in [16], then $\neg\Gamma^\bullet; \Delta^\bullet \vdash t : \neg A^\bullet$ holds in $\Lambda\mu_{\mathsf{cons}}$.*

Hence, every typable term in either $\lambda^{\rightarrow}$, $\lambda\mu$, or $\Lambda_{\mathcal{S}}$ is typable also in $\Lambda\mu_{\mathsf{cons}}$. On the other hand, due to the recursive stream types, there is a λ-term which is typable in $\Lambda\mu_{\mathsf{cons}}$, and not typable in $\lambda^{\rightarrow}$.

Gaboardi and Saurin [10] proposed another type system $\Lambda_{\mathcal{S}}$ as an extension of the type system in [15, 20], equipped with the recursive types and coercion operator from streams to terms, which enables to represent functions returning streams such as cdr. We can define a translation from $\Lambda\mu_{\mathsf{cons}}$ to $\Lambda_{\mathcal{S}}$ preserving typability.

References

1. R. Amadio and L. Cardelli. Subtyping recursive types. *ACM Transactions on Programming Languages and Systems*, 15(4):575–631, 1993.
2. Z.M. Ariola and J.W. Klop. Equational term graph rewriting. *Fundamenta Informaticae*, 26(3-4):207–240, 1996.
3. A. Carraro. The untyped stack calculus and Böhm's theorem. In *7th Workshop on Logical and Semantic Frameworks, with Applications (LSFA 2012)*, volume 113 of *Electric Proceedings in Theoretical Computer Science*, pages pp. 77–92, 2012.
4. A. Carraro, T. Ehrhard, and A. Salibra. The stack calculus. In *7th Workshop on Logical and Semantic Frameworks, with Applications (LSFA 2012)*, volume 113 of *Electric Proceedings in Theoretical Computer Science*, pages pp. 93–108, 2012.
5. R. David and W. Py. $\lambda\mu$-calculus and Böhm's theorem. *The Journal of Symbolic Logic*, 66:407–413, 2001.
6. P. de Groote. A CPS-translation of the $\lambda\mu$-calculus. In Tison, S., editor, *Proceedings of the Colloquium on Trees in Algebra and Programming (CAAP'94)*, volume 787 of *LNCS*, pages 85–99. Springer, 1994.
7. Dehornoy, P. and van Oostrom, V. Z, proving confluence by monotonic single-step upperbound functions, 2008.
8. U. de'Liguoro. The approximation theorem for the $\Lambda\mu$-calculus. unpublished manuscript, 2012. Available at `http://www.di.unito.it/~lambda/biblio/entry-ApproxLM12.html`.
9. H. Friedman. Equality between functionals. In Parikh, R., editor, *Logic Colloquium*, volume 453 of *Lecture Notes in Mathematics*, pages 22–37. Springer, 1973.
10. M. Gaboardi and A. Saurin. A foundational calculus for computing with streams. In *12th Italian Conference on Theoretical Computer Science*, 2010.
11. Komori, Y., Matsuda, N., and Yamakawa, F. A simplified proof of the church-rosser theorem. *Studia Logica*, 101(1), 2013.
12. R. Milner. A complete inference system for a class of regular begaviours. *Journal of Computer and System Sciences*, 28:439–466, 1984.
13. K. Nakazawa. Extensional models of typed lambda-mu caclulus. unpublished manuscript, 2014. Available at `http://www.fos.kuis.kyoto-u.ac.jp/~knak/...`
14. K. Nakazawa and S. Katsumata. Extensional models of untyped Lambda-mu calculus. In Geuvers, H. and de'Liguoro, U., editors, *Proceedings Fourth Workshop on Classical Logic and Computation (CL&C 2012)*, volume 97 of *Electric Proceedings in Theoretical Computer Science*, pages 35–47, 2012.

15. M. Pagani and A Saurin. Stream associative nets and $\lambda\mu$-calculus. Research Report 6431, INRIA, 2008.
16. M. Parigot. $\lambda\mu$-calculus: an algorithmic interpretation of classical natural deduction. In Voronkov, A., editor, *Proceedings of the International Conference on Logic Programming and Automated Reasoning (LPAR '92)*, volume 624 of *LNCS*, pages 190–201. Springer, 1992.
17. A. Salomaa. Two complete axiom systems for the algebra of regular events. *Journal of the Associations for Computing Machinery*, 13(1):158–169, 1966.
18. A. Saurin. Separation with streams in the $\Lambda\mu$-calculus. In *20th Annual IEEE Symposium on Logic in Computer Science (LICS' 05)*, pages 356–365, 2005.
19. A. Saurin. Standardization and Böhm trees for $\Lambda\mu$-calculus. In Blume, M., Kobayashi, N., and Vidal, G., editors, *Tenth International Symposium on Functional and Logic Programming (FLOPS 2010)*, volume 6009 of *LNCS*, pages 134–149. Springer, 2010.
20. A. Saurin. Typing streams in the $\Lambda\mu$-calculus. *ACM Transactions on Computational Logic*, 11, 2010.
21. T. Streicher and B. Reus. Classical logic, continuation semantics and abstract machines. *Journal of Functional Programming*, 8(6):543–572, November 1998.
22. H. Thielecke. *Categorical structure of continuation passing style.* PhD thesis, University of Edinburgh, 1997.
23. S. van Bakel, F. Barbanera, and U. de'Liguoro. 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. Springer, 2011.

A Equivalence on Stream Types

We give a direct proof of the following proposition. We use metavariables $\boldsymbol{\delta}$, $\boldsymbol{\delta'}$,$\cdots$ for sequences of term types. We write $|\boldsymbol{\delta}|$ for the length of the sequence $\boldsymbol{\delta}$, and write $\langle\!|\boldsymbol{\delta}|\!\rangle = \langle\!|\boldsymbol{\delta'}|\!\rangle$ for that $|\boldsymbol{\delta}| = |\boldsymbol{\delta'}|$ and $\langle\!|\delta_i|\!\rangle = \langle\!|\delta'_i|\!\rangle$ for any $0 \leq i \leq |\boldsymbol{\delta}| - 1$.

Proposition 10. *$\delta \sim \delta'$ ($\sigma \sim \sigma'$) iff $\langle\!|\delta|\!\rangle = \langle\!|\delta'|\!\rangle$ ($\langle\!|\sigma|\!\rangle = \langle\!|\sigma'|\!\rangle$, repectively).*

We divide the proposition into the two direction.

Proposition 11. *If $\delta \sim \delta'$ holds, then $\langle\!|\delta|\!\rangle = \langle\!|\delta'|\!\rangle$.*

Proof. By induction on $\delta \sim \delta'$.

Case (Fld). We have $\delta_0 \times \langle\!|\delta_1, \cdots, \delta_{n-1}, \delta_0|\!\rangle = \delta_0 \times \Pi_i \langle\!|\delta_{(i+1 \bmod n)}|\!\rangle = \Pi_i \langle\!|\delta_{(i \bmod n)}|\!\rangle$.

Case (Ctr). By the induction hypothesis, we have the following.

$$\langle\!|\delta_0|\!\rangle \times \cdots \times \langle\!|\delta_{n-1}|\!\rangle \times \langle\!|\sigma|\!\rangle = \langle\!|\sigma|\!\rangle \qquad (*)$$

We show that, for any $i \geq 0$, the i-th component of $\langle\!|\sigma|\!\rangle$ is $\langle\!|\delta_{i \bmod n}|\!\rangle$ by induction on i.

Subcase $0 \leq i < n$. By (*).

Subcase $i \geq n$. By (*), we have $\langle\!|\sigma_i|\!\rangle = \langle\!|\sigma_{i-n}|\!\rangle$. By the induction hypothesis, we have $\langle\!|\sigma_{i-n}|\!\rangle = \langle\!|\sigma_{i-n \bmod n}|\!\rangle = \langle\!|\sigma_{i \bmod n}|\!\rangle$.

In order to show the converse direction, we prove the following lemmas.

Lemma 6. *If $\langle\!|[\boldsymbol{\delta}]|\!\rangle = \langle\!|[\boldsymbol{\delta'}]|\!\rangle$ and $\gcd(|\boldsymbol{\delta}|, |\boldsymbol{\delta'}|) = k$, then there exists $\boldsymbol{\delta''}$ of the length k such that $\langle\!|\boldsymbol{\delta}|\!\rangle = \langle\!|\underbrace{\boldsymbol{\delta''}, \cdots, \boldsymbol{\delta''}}_{|\boldsymbol{\delta}|/k}|\!\rangle$ and $\langle\!|\boldsymbol{\delta'}|\!\rangle = \langle\!|\underbrace{\boldsymbol{\delta''}, \cdots, \boldsymbol{\delta''}}_{|\boldsymbol{\delta'}|/k}|\!\rangle$.*

Proof. Let $I = |\boldsymbol{\delta}|/k$ and $J = |\boldsymbol{\delta'}|/k$, $\boldsymbol{\delta} = \boldsymbol{\delta}_1, \cdots, \boldsymbol{\delta}_I$, and $\boldsymbol{\delta'} = \boldsymbol{\delta'}_1, \cdots, \boldsymbol{\delta'}_J$, where all of $|\boldsymbol{\delta}_i|$ and $|\boldsymbol{\delta'}_j|$ are k. Note that I and J are relatively prime.

For $l \in \mathbb{N}$, the components from $(l \cdot k)$-th to $((l+1) \cdot k - 1)$-th of $\langle\!|[\boldsymbol{\delta}]|\!\rangle$ and $\langle\!|[\boldsymbol{\delta'}]|\!\rangle$ are $\langle\!|\boldsymbol{\delta}_{l \bmod I}|\!\rangle$ and $\langle\!|\boldsymbol{\delta'}_{l \bmod J}|\!\rangle$, respectively. By the assumption $\langle\!|[\boldsymbol{\delta}]|\!\rangle = \langle\!|[\boldsymbol{\delta'}]|\!\rangle$, we have $\langle\!|\boldsymbol{\delta}_{l \bmod I}|\!\rangle = \langle\!|\boldsymbol{\delta'}_{l \bmod J}|\!\rangle$. By the Chinese remainder theorem, we have $\langle\!|\boldsymbol{\delta}_i|\!\rangle = \langle\!|\boldsymbol{\delta'}_j|\!\rangle$ for any i, j. Hence, let $\boldsymbol{\delta''} = \boldsymbol{\delta}_1$, and we have $\langle\!|\boldsymbol{\delta}|\!\rangle = \langle\!|\underbrace{\boldsymbol{\delta''}, \cdots, \boldsymbol{\delta''}}_{I}|\!\rangle$ and $\langle\!|\boldsymbol{\delta'}|\!\rangle = \langle\!|\underbrace{\boldsymbol{\delta''}, \cdots, \boldsymbol{\delta''}}_{J}|\!\rangle$.

Definition 11. *The depth of types are defined as follows.*

$$d(X) = 0 \qquad d(\sigma \to \delta) = \max\{d(\sigma), d(\delta)\} + 1$$
$$d([\delta_0, \cdots]) = \max\{d(\delta_0), \cdots\} + 1 \qquad d(\delta \times \sigma) = \max\{d(\delta) + 1, d(\sigma)\}$$

Lemma 7. *1. If $\delta \sim \delta'$ is derivable without the rule (Ctr), then $d(\delta) = d(\delta')$.*

2. If $\langle\!|\delta|\!\rangle = \langle\!|\delta'|\!\rangle$, then $d(\delta) = d(\delta')$.

Proof. 1. By induction on $\delta \sim \delta'$.

2. Define the depth of expanded types as

$$d(\mathit{\Pi}_{i\in\mathbb{N}}\delta_i) = \max\{d(\delta_i) \mid i \in \mathbb{N}\} + 1,$$

and then we have $d(\sigma) = d(\langle\!|\sigma|\!\rangle)$.

Proposition 12. *If $\langle\!|\delta|\!\rangle = \langle\!|\delta'|\!\rangle$ holds, then $\delta \sim \delta'$.*

Proof. We prove $\langle\!|\delta|\!\rangle = \langle\!|\delta'|\!\rangle \Rightarrow \delta \sim \delta'$ and $\langle\!|\sigma|\!\rangle = \langle\!|\sigma'|\!\rangle \Rightarrow \sigma \sim \sigma'$ simultaneously by induction on $d(\delta)$ and $d(\sigma)$. We will see the second statement.

For any stream types σ and σ', we can derive the following without (Ctr) for some δ_i, δ'_i, $\boldsymbol{\delta}$, and $\boldsymbol{\delta}'$:

$$\sigma \sim \delta_1 \times \cdots \times \delta_m \times [\boldsymbol{\delta}] \qquad\qquad \sigma' \sim \delta'_1 \times \cdots \times \delta'_m \times [\boldsymbol{\delta}'].$$

By the assumption, we have $\langle\!|\delta_i|\!\rangle = \langle\!|\delta'_i|\!\rangle$ for $1 \le i \le m$ and $\langle\!|[\boldsymbol{\delta}]|\!\rangle = \langle\!|[\boldsymbol{\delta}']|\!\rangle$. By Lemma 7, $d(\delta_i)$ for any i and $d(\delta)$ for $\delta \in \boldsymbol{\delta}$ are less than $d(\sigma)$. Hence we have $\delta_i \sim \delta'_i$ by the induction hypothesis. By Lemma 6, there exists $\boldsymbol{\delta}''$ such that $\langle\!|\boldsymbol{\delta}|\!\rangle = \langle\!|\boldsymbol{\delta}'', \cdots, \boldsymbol{\delta}''|\!\rangle$ and $\langle\!|\boldsymbol{\delta}'|\!\rangle = \langle\!|\boldsymbol{\delta}'', \cdots, \boldsymbol{\delta}''|\!\rangle$, and we have $\boldsymbol{\delta} \sim (\boldsymbol{\delta}'', \cdots, \boldsymbol{\delta}'')$ and $\boldsymbol{\delta}' \sim (\boldsymbol{\delta}'', \cdots, \boldsymbol{\delta}'')$ by the induction hypothesis. Therefore, we have $[\boldsymbol{\delta}] \sim [\boldsymbol{\delta}''] \sim [\boldsymbol{\delta}']$, and then $\sigma \sim \sigma'$.

B Proof of Subject Reduction

Lemma 8. *1. $\delta \sim \sigma' \to \delta' \Leftrightarrow \exists\sigma''\delta''.[\delta = \sigma'' \to \delta'' \,\&\, \sigma' \sim \sigma'' \,\&\, \delta' \sim \delta'']$*

2. $\delta \sim \delta' \times \sigma' \Leftrightarrow \delta \sim \delta' \,\&\, \sigma \sim \sigma'$

Proof. $\Leftarrow$ in both 1 and 2 are trivial, and $\Rightarrow$ is proved by Proposition 6.

Lemma 9 (Generation lemma). *1. $\mathit{\Gamma} \mid \mathit{\Delta} \vdash x : \delta \Rightarrow \exists(x : \delta') \in \mathit{\Gamma}.[\delta \sim \delta']$*

2. $\mathit{\Gamma} \mid \mathit{\Delta} \vdash \lambda x.t : \delta \Rightarrow \exists\delta'\sigma'\delta''.[\mathit{\Gamma}, x : \delta' \mid \mathit{\Delta} \vdash t : \sigma' \to \delta'' \,\&\, \delta \sim \delta' \times \sigma' \to \delta'']$

3. $\mathit{\Gamma} \mid \mathit{\Delta} \vdash tu : \delta \Rightarrow \exists\delta'\sigma'\delta''.[\mathit{\Gamma} \mid \mathit{\Delta} \vdash t : \delta' \times \sigma' \to \delta'' \,\&\, \mathit{\Gamma} \mid \mathit{\Delta} \vdash u : \delta' \,\&\, \delta = \sigma' \to \delta'']$

4. $\mathit{\Gamma} \mid \mathit{\Delta} \vdash \mu\alpha.t : \delta \Rightarrow \exists\sigma'\delta'.[\mathit{\Gamma} \mid \mathit{\Delta}, \alpha : \sigma' \vdash t : \delta' \,\&\, \delta \sim \sigma' \to \delta']$

5. $\mathit{\Gamma} \mid \mathit{\Delta} \vdash tS : \delta \Rightarrow \exists\sigma'.[\mathit{\Gamma} \mid \mathit{\Delta} \vdash t : \sigma' \to \delta \,\&\, \mathit{\Gamma} \mid \mathit{\Delta} \vdash S : \sigma']$

6. $\mathit{\Gamma} \mid \mathit{\Delta} \vdash \mathsf{car}S : \delta \Rightarrow \exists\sigma'.[\mathit{\Gamma} \mid \mathit{\Delta} \vdash S : \delta \times \sigma']$

7. $\mathit{\Gamma} \mid \mathit{\Delta} \vdash \alpha : \sigma \Rightarrow \exists(\alpha : \sigma') \in \mathit{\Delta}.[\sigma \sim \sigma']$

8. $\mathit{\Gamma} \mid \mathit{\Delta} \vdash t :: S : \sigma \Rightarrow \exists\delta'\sigma'.[\mathit{\Gamma} \mid \mathit{\Delta} \vdash t : \delta' \,\&\, \mathit{\Gamma} \mid \mathit{\Delta} \vdash S : \sigma' \,\&\, \sigma \sim \delta' \times \sigma']$

9. $\mathit{\Gamma} \mid \mathit{\Delta} \vdash \mathsf{cdr}S : \sigma \Rightarrow \exists\delta'.[\mathit{\Gamma} \mid \mathit{\Delta} \vdash S : \delta' \times \sigma]$

Proof. By induction on the type derivation.

Lemma 10 (Substitution lemma). *1. $\mathit{\Gamma}, x : \delta \mid \mathit{\Delta} \vdash t : \delta' \,\&\, \mathit{\Gamma} \mid \mathit{\Delta} \vdash u : \delta \Rightarrow \mathit{\Gamma} \mid \mathit{\Delta} \vdash t[x := u] : \delta'$*

2. $\mathit{\Gamma} \mid \mathit{\Delta}, \alpha : \sigma \vdash t : \delta \,\&\, \mathit{\Gamma} \mid \mathit{\Delta} \vdash S : \sigma \Rightarrow \mathit{\Gamma} \mid \mathit{\Delta} \vdash t[\alpha := S] : \delta$

Proof. By induction on the type derivation of t.

Lemma 11. *Suppose $\Gamma \mid \Delta \vdash S : \delta_0 \times \cdots \times \delta_n \times \sigma$ and $\Gamma \mid \Delta \vdash S : \delta'_0 \times \cdots \times \delta'_n \times \sigma'$. Let δ''_i be either δ_i or δ'_i $(0 \leq i \leq n)$and σ'' be either σ or σ'. Then, we have $\Gamma \mid \Delta \vdash S : \delta''_0 \times \cdots \times \delta''_n \times \sigma''$.*

Proof. By induction on S.

Case $S \equiv \alpha$. By the generation lemma, we have $\alpha : \sigma_\alpha \in \Delta$ such that $\sigma_\alpha \sim \delta_0 \times \cdots \times \delta_n \times \sigma \sim \delta'_0 \times \cdots \times \delta'_n \times \sigma'$. Then, we have $\delta_i \sim \delta'_i$ $(0 \leq i \leq n)$ and $\sigma \sim \sigma'$. Therefore, we have $\sigma_\alpha \sim \delta''_0 \times \cdots \times \delta''_n \times \sigma''$.

Case $S \equiv t' :: S'$. By Lemma 8 and the generation lemma, we have

$$\begin{array}{ll} \Gamma \mid \Delta \vdash t' : \delta_0 & \Gamma \mid \Delta \vdash S' : \delta_1 \times \cdots \delta_n \times \sigma \\ \Gamma \mid \Delta \vdash t' : \delta'_0 & \Gamma \mid \Delta \vdash S' : \delta'_1 \times \cdots \delta'_n \times \sigma'. \end{array}$$

By the induction hypothesis, we have $\Gamma \mid \Delta \vdash S' : \delta''_1 \times \cdots \delta'_n \times \sigma''$, and hence $\Gamma \mid \Delta \vdash t' :: S' : \delta''_0 \times \cdots \delta'_n \times \sigma''$.

Case $S \equiv \mathsf{cdr}S'$. By the generation lemma, we have $\Gamma \mid \Delta \vdash S : \delta \times \delta_0 \times \cdots \times \delta_n \times \sigma$ and $\Gamma \mid \Delta \vdash S : \delta' \times \delta'_0 \times \cdots \times \delta'_n \times \sigma'$ for some δ and δ'. For any δ''_i $(0 \leq i \leq n)$ and σ'', we have $\Gamma \mid \Delta \vdash S : \delta \times \delta''_0 \times \cdots \times \delta''_n \times \sigma''$, and hence $\Gamma \mid \Delta \vdash \mathsf{cdr}S : \delta''_0 \times \cdots \times \delta''_n \times \sigma''$.

Theorem 9 (Subject reduction). *If $\Gamma \mid \Delta \vdash t : \delta$ and $t \to u$ hold, then we have $\Gamma \mid \Delta \vdash u : \delta$.*

Proof. Induction on $t \to u$.

Case (β_T). Suppose $\Gamma \mid \Delta \vdash (\mu\alpha.t)u : \delta$. By the generation lemma, we have $\Gamma \mid \Delta, \alpha : \sigma'' \vdash t : \delta''$, $\Gamma \mid \Delta \vdash u : \delta'$, $\sigma'' \sim \delta' \times \sigma'$, and $\delta = \sigma' \to \delta''$. Then, we have $\Gamma \mid \Delta, \alpha : \sigma' \vdash u :: \alpha : \delta' \times \sigma'(\sim \sigma'')$. By the substitution lemma, we have $\Gamma \mid \Delta, \alpha : \sigma' \vdash t[\alpha := u :: \alpha] : \delta'$, and hence $\Gamma \mid \Delta \vdash \mu\alpha.t[\alpha := u :: \alpha] : \sigma' \to \delta'(= \delta)$.

Case $(\eta_{::})$. Suppose $\Gamma \mid \Delta \vdash (\mathsf{car}S) :: (\mathsf{cdr}S) : \sigma$. By the generation lemma, we have $\Gamma \mid \Delta \vdash \mathsf{car}S : \delta'$ and $\Gamma \mid \Delta \vdash \mathsf{cdr}S : \sigma'$ for $\sigma \sim \delta' \times \sigma'$. By the generation lemma again, we have $\Gamma \mid \Delta \vdash S : \delta' \times \sigma_1$ for some σ_1 and $\Gamma \mid \Delta \vdash S : \delta_2 \times \sigma'$ for some δ_2. By Lemma 11, we have $\Gamma \mid \Delta \vdash S : \delta' \times \sigma'$.

The other cases are similarly proved.

C Proof of Lemma 4

Lemma 12. *For any $C \in \mathsf{SN}^\beta_\mathsf{C}$, we have $C[x] \in \mathsf{SN}^\beta_\mathsf{T}$ and $C[\mathsf{cadr}^n\alpha] \in \mathsf{SN}^\beta_\mathsf{T}$.*

Proof. In this proof, the reduction defined by only (assoc) is denoted by $\to_\mathsf{a}$, and $\to_\mathsf{na}$ is defined as $\to_\beta \setminus \to_\mathsf{a}$. We define contexts as $C ::= [] \mid Ct \mid CS$. For any context C, it is easy to see that

(i) $C[x] \to_\beta u$ implies $u = C'[x]$ for some context C', and

(ii) $C[x] \to_\mathsf{a} \cdot \to_\mathsf{na} C''[x]$ implies $C[x] \to_\mathsf{na} \cdot \to_\mathsf{a} C''[x]$.

The only nontrivial case for (ii) is that $C[x] = C_0[C_1[x](t :: S)] \to_\mathsf{a} C_0[C_1[x]tS] \to_\mathsf{na} C_0[C_1[x]t'S'] = C''[x]$. Then, we have $C[x] \to_\mathsf{na} C_0[C_1[x](t' ::$

$S')] \to_{\mathsf{a}} C''[x]$. Since $\to_{\mathsf{a}}$ is strongly normalizable, if there exists an infinite β-reduction sequence from $C[x]$, then we have an infinite sequence of $\to_{\mathsf{na}}$ from $C[x]$. If $C \in \mathsf{SN}^{\beta}_{\mathsf{S}}$ and there is an infinite sequence of $\to_{\mathsf{na}}$ from $C[x]$, there is an infinite reduction sequence from one of the stream arguments in C, that contradicts.

For $C[\mathsf{cadr}^n\alpha]$, it is similarly proved.

Proof (of Lemma 4). They are simultaneously proved by structural induction on the expanded types.

Case X. 1 is trivial. 2 follows from Lemma 12.

Case $\sigma \to \delta$. For 1, suppose that $t \in \mathsf{Red}_{\sigma\to\delta}$. By the induction hypothesis for 3, we have $\alpha \in \mathsf{Red}_{\sigma}$, so $t\alpha \in \mathsf{Red}_{\delta}$. By the induction hypothesis for 1, we have $\mathsf{Red}_{\delta} \subseteq \mathsf{SN}^{\beta}_{\mathsf{T}}$, so $t\alpha \in \mathsf{SN}^{\beta}_{\mathsf{T}}$. Hence $t \in \mathsf{SN}^{\beta}_{\mathsf{T}}$. For $S \in \mathsf{Red}_{\sigma}$, we have $\mathsf{car}S \in \mathsf{Red}_{(\sigma)_0} \subseteq \mathsf{SN}^{\beta}_{\mathsf{T}}$, and hence $S \in \mathsf{SN}^{\beta}_{\mathsf{S}}$.

For 2, let $C \in \mathsf{SN}^{\beta}_{\mathsf{C}}$ and $S \in \mathsf{Red}_{\sigma}$. By the induction hypothesis for 1, we have $S \in \mathsf{SN}^{\beta}_{\mathsf{S}}$, so $CS \in \mathsf{SN}^{\beta}_{\mathsf{C}}$. By the induction hypothesis for 2, we have $C[x]S \in \mathsf{Red}_{\delta}$. $C[\mathsf{cadr}^n\alpha] \in \mathsf{Red}_{\sigma\to\delta}$ is similarly proved.

Case $\mathit{\Pi}_n\delta_n$. We have $\mathsf{cadr}^n\alpha \in \delta_n$ for any $n$by the induction hypothesis for 2.

D Separation theorem for $\Lambda\mu_{\mathsf{cons}}$

We give the proofs of conservativity of $\Lambda\mu_{\mathsf{cons}}$ over $\Lambda\mu$ and the separation theorem of $\Lambda\mu_{\mathsf{cons}}$.

Definition 12. *The translation t° from $\Lambda\mu_{\mathsf{cons}}$ to $\Lambda\mu$ is defined as follows.*

$$\begin{aligned} x^\circ &\equiv x & t \circledast \alpha &\equiv t\alpha \\ (\lambda x.t)^\circ &\equiv \lambda x.t^\circ & t \circledast (u :: S) &\equiv tu^\circ \circledast S \\ (tu)^\circ &\equiv t^\circ u^\circ & t \circledast (\mathsf{cdr}S) &\equiv K_0 t \circledast S \\ (\mu\alpha.t)^\circ &\equiv \mu\alpha.t^\circ \\ (tS)^\circ &\equiv t^\circ \circledast S \\ (\mathsf{car}S)^\circ &\equiv K_1 \circledast S \end{aligned}$$

Lemma 13. *1. For any $\Lambda\mu$-term t, we have $t^\circ \equiv t$.*

2. For any stream closed t and u, if $t =_{\Lambda\mu_{\mathsf{cons}}} u$, then $t^\circ = u^\circ$ holds in $\Lambda\mu$.

Proof. 1. By induction on $\Lambda\mu$-terms t.

2. For a $\Lambda\mu$-term t, the *t-closure context* is defined as a context C such that $C[t]$ is stream closed. Then, we can show the following:

(i) if $t =_{\Lambda\mu_{\mathsf{cons}}} u$ holds, then $C[t^\circ] = C[t'^\circ]$ holds in $\Lambda\mu$ for any tt'-closure context C,

(ii) if $S =_{\Lambda\mu_{\mathsf{cons}}} S'$ holds , then $C[t \circledast S] = C[t \circledast S']$ holds in $\Lambda\mu$ for any $\Lambda\mu$-term t and any tSS'-closure context C.

Since t and u are stream closed, [] is a tu-closure context, and hence we have $t^\circ =_{\Lambda\mu} u^\circ$.

Proposition 13 (Conservativity). *For any stream closed $\Lambda\mu$-terms t and u, $t = u$ holds in $\Lambda\mu_{\mathsf{cons}}$ if and only if $t = u$ holds in $\Lambda\mu$.*

Proof. The if part is easily proved by induction on $t =_{\Lambda\mu_{\mathsf{cons}}} u$. For the only-if part, suppose that t and u are $\Lambda\mu$-terms and $t =_{\Lambda\mu_{\mathsf{cons}}} u$ holds. By Lemma 13, we have $t \equiv t^\circ =_{\Lambda\mu} u^\circ \equiv u$.

Corollary 4. *For any stream closed $\Lambda\mu$-terms t and u, $t = u$ holds in the $\Lambda\mu$-calculus if and only if $[\![t]\!]_\rho = [\![u]\!]_\rho$ for any stream model $\mathcal{A}$ and ρ.*

Proof. Soundness has been already proved. Conversely, if $[\![t]\!]_\rho = [\![u]\!]_\rho$ holds for the term model of $\Lambda\mu_{\mathsf{cons}}$ and the canonical environment ρ as in the proof of Theorem 1, then we have $t = u$ in $\Lambda\mu_{\mathsf{cons}}$. By the conservativity, $t = u$ holds in $\Lambda\mu$.

The following is the proof of the separation theorem for $\Lambda\mu_{\mathsf{cons}}$.

Lemma 14. *For any $\Lambda\mu_{\mathsf{cons}}$-term t, we have $t =_{\Lambda\mu_{\mathsf{cons}}} t^\circ$. In particular, if t is stream closed and $\to_\beta$-normal in $\Lambda\mu_{\mathsf{cons}}$, then u has a canonical normal form in $\Lambda\mu$.*

Proof. $t =_{\Lambda\mu_{\mathsf{cons}}} t^\circ$ is proved by induction on t. Suppose that a $\Lambda\mu_{\mathsf{cons}}$-term t is $\to_\beta$-normal. Since we have $(u \circledast \mathsf{cdr}^n\alpha)^\circ = (K_0^n u)\alpha$ for any $\Lambda\mu$-term u, t° belongs to the subset of the $\Lambda\mu$-terms defined as

$$t ::= a \mid \mu\alpha.t \qquad\qquad a ::= x \mid (K_0^n K_1)\alpha \mid at \mid (K_0^n a)\alpha.$$

Applying (*fst*) for sufficiently many times, we have some u such that $t^\circ \to_{fst} u$ and u belongs to the subset of the $\Lambda\mu$-terms defined as

$$t ::= a \mid \lambda x.t \mid \mu\alpha.t \qquad a ::= x \mid (K_0^n K_1)x_0 \cdots x_m\alpha \mid at \mid (K_0^n a)x_0 \cdots x_m\alpha,$$

where $0 \leq n \leq m$. Since we have $(K_0^n K_1)x_0 \cdots x_m\alpha \to^*_{\Lambda\mu} x_n$ and $(K_0^n a)x_0 \cdots x_m\alpha \to^*_{\Lambda\mu} ax_n \cdots x_m\alpha$, and hence u has a canonical normal form, so does t°.

Theorem 10 (Separation). *Suppose that t_1 and t_2 are distinct stream closed normal $\Lambda\mu_{\mathsf{cons}}$-terms For any normal $\Lambda\mu_{\mathsf{cons}}$-terms u_1 and u_2, there exists an applicative context C such that $C[t_1] \to^* u_1$ and $C[t_2] \to^* u_2$ hold in $\Lambda\mu_{\mathsf{cons}}$, where the applicative contexts are defined as $C ::= [] \mid Ct \mid C\alpha$.*

Proof. By the confluence, we have $t_1 \neq_{\Lambda\mu_{\mathsf{cons}}} t_2$. By Lemma 14, we have canonical normal $\Lambda\mu$-terms t'_1 and t'_2 such that $t_i =_{\Lambda\mu_{\mathsf{cons}}} t'_i$ $(i = 1, 2)$. We also have $t'_1 \neq_{\Lambda\mu_{\mathsf{cons}}} t'_2$, and hence $t'_1 \neq_{\Lambda\mu} t'_2$ by the conservativity. By the separation theorem of $\Lambda\mu$, there exists an applicative context C such that $C[t'_1] =_{\Lambda\mu} \lambda xy.x$ and $C[t'_2] =_{\Lambda\mu} \lambda xy.y$, and they also hold in $\Lambda\mu_{\mathsf{cons}}$. Therefore, for the applicative context $C' = Cu_1u_2$, we have $C'[t'_i] =_{\Lambda\mu_{\mathsf{cons}}} u_i$ $(i = 1, 2)$, and hence we have $C'[t'_i] \to^* u_i$ in $\Lambda\mu_{\mathsf{cons}}$ by the confluence.