Extensional Models of Typed Lambda-mu Calculus

Koji Nakazawa

Graduate School of Informatics, Kyoto University

Abstract. This paper shows that the stream models of Nakazawa and Katsumata can be adapted to a typed setting for an extension of the $\Lambda\mu$-calculus, called $\Lambda\mu_{\mathsf{cons}}$. It shows the typed $\Lambda\mu_{\mathsf{cons}}$ is sound and complete with respect to the stream models. It also shows that any individual stream model with whole function spaces and infinite bases characterizes the extensional equality. This result corresponds to Friedman's theorem for the simply-typed λ-calculus and the full type hierarchies.
Keywords: extensional models, Lambda-mu calculus, Friedman's theorem

1 Type System for $\Lambda\mu_{\mathsf{cons}}$

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 [15] and [4] that it is a function application of t to a stream α.

Definition 1 (Terms and streams). 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

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

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*.

The type assignment system for $\Lambda\mu_{\mathsf{cons}}$ given here is inspired by the type system for the $\Lambda\mu$-calculus of de'Liguoro in [6], and adopting recursive types to represent types of streams like [8].

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 [14, 9, 1], and [2].

Definition 2 (Type assignment for $\Lambda\mu_{\mathsf{cons}}$). 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.

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

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.

$$\frac{}{\Gamma, x : \delta \mid \Delta \vdash x : \delta}\ (\mathrm{Ax}) \quad \frac{}{\Gamma \mid \Delta, \alpha : \sigma \vdash \alpha : \sigma}\ (\mathrm{SAx})$$

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

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

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

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

In the following, we suppose that each variable is assigned a fixed type. The variable x assigned the type δ is denoted by x^δ. The variables x^δ and $x^{\delta'}$ are considered as distinct variables when δ and δ' are different types. For terms and streams, we also use the annotation such as t^δ and S^σ to make their types explicit.

Definition 3 (Extensional equality). *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. The relation $\Gamma \mid \Delta \vdash t_1 = t_2 : \delta$ means that $\Gamma \mid \Delta \vdash t_i : \delta$ $(i = 1, 2)$ and $t_1 =_{\Lambda\mu_{\mathsf{cons}}} t_2$ hold. For simplicity, we write just $t_1 =_{\Lambda\mu_{\mathsf{cons}}} t_2$ to mean $\Gamma \mid \Delta \vdash t_1 = t_2 : \delta$ for some Γ, Δ, and δ.

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 $(0 \leq i \leq n)$.

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

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

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 [8]. 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 4. *1. The expanded types are defined by*

$$\delta ::= X \mid \sigma \to \delta \qquad\qquad \sigma ::= \Pi_{i\in\mathbb{N}}\delta_i$$

We use the notation $\delta_0 \times \cdots \times \delta_{n-1} \times \Pi_{i\in\mathbb{N}}\delta'_i$ also for the expanded types to denote the infinite product $\Pi_{i\in\mathbb{N}}\delta''_i$ where

$$\delta''_i = \begin{cases} \delta_i & 0 \leq i < n \\ \delta'_{i-n} & i \geq n \end{cases}$$

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 $(\Pi_{j\in\mathbb{N}}\delta_j)_i = \delta_i$.

3. The expansion of the types is the following function from the types to the expanded types.

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

Note that the relation $\sqsubset$ defined as

$$\sigma, \delta \sqsubset \sigma \to \delta \qquad\qquad \delta_i \sqsubset \Pi_{i\in\mathbb{N}}\delta_i$$

is a well-founded order, and we will use the induction on this order.

Proposition 1. *$\delta \sim \delta'$ ($\sigma \sim \sigma'$) iff $(\!|\delta|\!) = (\!|\delta'|\!)$ ($(\!|\sigma|\!) = (\!|\sigma'|\!)$, respectively).*

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 1. In [10], 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.

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 considered in [17], 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 4.

As we will discuss in Section 4, the types of $\Lambda\mu_{\mathsf{cons}}$ can be seen as a restriction of the types of $\Lambda_{\mathcal{S}}$ of [12], but, regarding typability, $\Lambda\mu_{\mathsf{cons}}$ is not less than $\Lambda_{\mathcal{S}}$. Furthermore, it has some fundamental properties such as subject reduction and strong normalization for a reduction system for $\Lambda\mu_{\mathsf{cons}}$ [11].

2 Stream Models for Typed $\Lambda\mu_{\mathsf{cons}}$

The idea of the stream models is adapted to the typed $\Lambda\mu_{\mathsf{cons}}$.

In the following, we consider families of sets $\mathcal{A}^\delta$ indexed by the expanded types, and the terms of type δ is interpreted as elements in $\mathcal{A}^{(\!|\delta|\!)}$. For simplicity, we do not distinguish the notation for the types and the expanded types, using the notation $\mathcal{A}^\delta$ for $\mathcal{A}^{(\!|\delta|\!)}$.

Definition 5. 1. A *stream pre-model* $\mathcal{A}$ for the typed $\Lambda\mu_{\mathsf{cons}}$ consists of the following:

- $\{\mathcal{A}^\delta\}_\delta$ and $\{\mathcal{A}^\sigma\}_\sigma$: families of sets indexed by the expanded types.
- A function $(\star) : \mathcal{A}^{\sigma\to\delta} \times \mathcal{A}^\sigma \to \mathcal{A}^\delta$ 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 $(::)$ from $\mathcal{A}^\delta \times \mathcal{A}^\sigma$ to $\mathcal{A}^{\delta\times\sigma}$ for each δ and σ, the inverse of which consists of the projection functions $\langle \mathsf{Car}, \mathsf{Cdr}\rangle$. Then, we define the i-th projection function $\pi_i : \mathcal{A}^\sigma \to \mathcal{A}^{(\sigma)_i}$ for each $i \geq 0$ as $\pi_i(s) = \mathsf{Car}(\mathsf{Cdr}^i s)$.

2. For a stream pre-model $\mathcal{A}$, an *environment* on $\mathcal{A}$ is a map from variables to $\mathcal{A}$ such that $\rho(x^\delta) \in \mathcal{A}^\delta$ for each $x^\delta \in \mathsf{TV}$ and $\rho(\alpha^\sigma) \in \mathcal{A}^\sigma$ for each $\alpha^\sigma \in \mathsf{SV}$. The set of environments on $\mathcal{A}$ is denoted by $\mathsf{Env}^{\mathcal{A}}$.

3. A *stream model* $\mathcal{A}$ is a stream pre-model equipped with the *meaning function* $[\![\cdot]\!]$ such that the following hold:

– $[\![x^\delta]\!]_\rho = \rho(x^\delta)$ and $[\![\alpha^\sigma]\!]_\rho = \rho(\alpha^\sigma)$.
– $[\![\lambda x^{\delta'}.t^{\sigma\to\delta}]\!]_\rho \in \mathcal{A}^{\delta'\times\sigma\to\delta}$, and $[\![\lambda x^{\delta'}.t^{\sigma\to\delta}]\!]_\rho \star s = [\![t]\!]_{\rho[x\mapsto \mathsf{Car}(s)]} \star \mathsf{Cdr}(s)$ for any $s \in \mathcal{A}^{\delta'\times\sigma}$,
– $[\![t^{\delta'\times\sigma\to\delta} u^{\delta'}]\!]_\rho \in \mathcal{A}^{\sigma\to\delta}$, and $[\![t^{\delta'\times\sigma\to\delta} u^{\delta'}]\!]_\rho \star s = [\![t]\!]_\rho \star ([\![u]\!]_\rho :: s)$ for any $s \in \mathcal{A}^\sigma$,
– $[\![\mu\alpha^\sigma.t^\delta]\!]_\rho \in \mathcal{A}^{\sigma\to\delta}$, and $[\![\mu\alpha^\sigma.t^\delta]\!]_\rho \star s = [\![t]\!]_{\rho[\alpha\mapsto s]}$ for any $s \in \mathcal{A}^\sigma$,
– $[\![t^{\sigma\to\delta} S^\sigma]\!]_\rho = [\![t]\!]_\rho \star [\![S]\!]_\rho$.
– $[\![t^\delta :: S^\sigma]\!]_\rho = [\![t]\!]_\rho :: [\![S]\!]_\rho$.
– $[\![\mathsf{cdr} S^{\delta\times\sigma}]\!]_\rho = \mathsf{Cdr}[\![S]\!]_\rho$.

4. A stream pre-model $\mathcal{A}$ 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.

For example, let $\mathcal{A}^X$ be an arbitrary non-empty set for each X, and then we can define a full stream model by $\mathcal{A}^{\sigma\to\delta} = \mathcal{A}^\sigma \to \mathcal{A}^\delta$, which is the whole function space, and $\mathcal{A}^\sigma = \Pi_{i\in\mathbb{N}}\mathcal{A}^{(\sigma)_i}$. An example of non-full stream model is the pointed CPO model, in which $\mathcal{A}^{\sigma\to\delta}$ is the set of the continuous functions and $\mathcal{A}^\sigma$ is similarly defined as $\Pi_{i\in\mathbb{N}}\mathcal{A}^{(\sigma)_i}$.

In these two models, $\mathcal{A}^\sigma$ is canonically defined as $\Pi_{i\in\mathbb{N}}\mathcal{A}^{(\sigma)_i}$, but $\mathcal{A}^\sigma$ is not necessarily isomorphic to $\Pi_i\mathcal{A}^{(\sigma)_i}$ in general. One example is the following open term model, in which, from an infinite tuple $\langle t_0, t_1, \cdots\rangle$, we cannot construct a stream S such that $\mathsf{cadr}_i(S) =_{\Lambda\mu_{\mathsf{cons}}} t_i$ for any $i \geq 0$. Moreover, the mapping $\langle \pi_i\rangle_{i\in\mathbb{N}}$ from $\mathcal{A}^\sigma$ to $\Pi_{i\in\mathbb{N}}\mathcal{A}^{(\sigma)_i}$ is not necessarily injective. However, we can assume injectivity without any change of the following discussion, since the mapping $[S] \mapsto \langle [\mathsf{cadr}_i(S)]\rangle_{i\in\mathbb{N}}$ is injective in the open term model (Lemma 1).

Proposition 2 (Open term model). *Define the sets*

$$[t] = \{u \mid \exists \Gamma, \Delta, \delta.[\Gamma; \Delta \vdash t = u : \delta]\} \qquad \mathcal{T}^\delta = \{[t] \mid \exists \Gamma, \Delta.[\Gamma; \Delta \vdash t : \delta]\}$$
$$[S] = \{U \mid \exists \Gamma, \Delta, \sigma.[\Gamma; \Delta \vdash T = U : \sigma]\} \quad \mathcal{T}^\sigma = \{[S] \mid \exists \Gamma, \Delta.[\Gamma; \Delta \vdash S : \sigma]\},$$

and the functions

$$[t]\star[S] = [tS] \qquad [t] :: [S] = [t :: S]$$
$$\mathsf{Car}[S] = [\mathsf{car}\, S] \qquad \mathsf{Cdr}[S] = [\mathsf{cdr} S].$$

This structure $\mathcal{T}$ is a stream model with the meaning function given by $[\![t]\!]_\rho = [t\theta_\rho]$, where $\theta_\rho(x) = u$ for $\rho(x) = [u]$ and $\theta_\rho(\alpha) = S$ for $\rho(\alpha) = [S]$.

Proof. Note that the functions, and $[t\theta_\rho]$ are well-defined. Bijectivity of $(::)$ follows from the axiom (surj), and extensionality of $(\star)$ follows from the axiom (η_S). The conditions for the meaning function are proved similarly to the untyped case.

Theorem 1 (Soundness and completeness). *For any t and u, $\Gamma; \Delta \vdash t = u : \delta$ holds for some Γ, Δ, and δ if and only if $[\![t]\!]_\rho = [\![u]\!]_\rho$ holds for any stream model $\mathcal{A}$ and $\rho \in \mathsf{Env}^{\mathcal{A}}$.*

Proof. $(\Rightarrow)$ The soundness is proved by induction on $\Gamma; \Delta \vdash t = u : \delta$.

$(\Leftarrow)$ For the open term model, take an environment ρ as $\rho(x) = [x]$ and $\rho(\alpha) = [\alpha]$. By the assumption, we have $[\![t]\!]^{\mathcal{T}}_\rho = [\![u]\!]^{\mathcal{T}}_\rho$, so we have $t =_{\Lambda\mu_{\mathsf{cons}}} u$.

One conclusion of the soundness (and the existence of a non-trivial stream model) is that $(\mathsf{cdr}^n\alpha)^\sigma =_{\Lambda\mu_{\mathsf{cons}}} (\mathsf{cdr}^m\beta)^\sigma$ if and only if $\alpha \equiv \beta$ and $n = m$. Indeed, if either $\alpha \not\equiv \beta$ or $n \neq m$, we can construct a stream model which distinguishes the two streams $\mathsf{cdr}^n\alpha$ and $\mathsf{cdr}^m\beta$.

Another conclusion is the consistency of $\Lambda\mu_{\mathsf{cons}}$.

Theorem 2 (Consistency). *There exist closed $\Lambda\mu_{\mathsf{cons}}$-terms t and u such that $;\vdash t : \delta$ and $;\vdash u : \delta$ hold, and $;\vdash t =_{\Lambda\mu_{\mathsf{cons}}} u$ does not hold.*

Proof. As the proof of the previous lemma, we have a non-trivial stream model containing two distinct elements, so we have $\lambda xy.x \neq \lambda xy.y$ in a similar way to the untyped case.

3 Friedman's Theorem for Typed $\Lambda\mu_{\mathsf{cons}}$

In this subsection, we will define the logical relations on the stream models for the typed $\Lambda\mu_{\mathsf{cons}}$, and by means of them, a characterization of the equality in $\Lambda\mu_{\mathsf{cons}}$ by a full stream model will be given.

First, we prove the injectivity of $\langle \mathsf{cadr}_i \rangle_{i\in\mathbb{N}}$ by the soundness and the existence of a non-trivial stream model.

Lemma 1. *For any streams S_1 and S_2, if $\mathsf{cadr}_i(S_1) =_{\Lambda\mu_{\mathsf{cons}}} \mathsf{cadr}_i(S_2)$ holds for all $i \geq 0$, then we have $S_1 =_{\Lambda\mu_{\mathsf{cons}}} S_2$.*

Proof. We can easily show, by the induction on S, that every S can be represented as the form of $S =_{\Lambda\mu_{\mathsf{cons}}} t_1 :: \cdots :: t_n :: \mathsf{cdr}^m\alpha$ for some n, m, α, and t_i $(1 \leq i \leq n)$. Then, we can assume $S_1 = t_1 :: \cdots :: t_n :: \mathsf{cdr}^m\alpha$ and $S_2 = u_1 :: \cdots :: u_n :: \mathsf{cdr}^k\beta$ for some n, m, and k, since we have $\mathsf{cdr}^i\gamma =_{\Lambda\mu_{\mathsf{cons}}} (\mathsf{carcdr}^i\gamma) :: \mathsf{cdr}^{i+1}\gamma$ by (surj). By the assumption, we have $t_i =_{\Lambda\mu_{\mathsf{cons}}} u_i$ for any $1 \leq i \leq n$. If $\alpha \not\equiv \beta$, we can define ρ on a non-trivial stream model $\mathcal{A}$ such that $\pi_m(\rho(\alpha)) \neq \pi_k(\rho(\beta))$. For this ρ, we have $[\![\mathsf{cadr}_m(\alpha)]\!]^{\mathcal{A}}_\rho = \pi_m(\rho(\alpha)) \neq \pi_k(\rho(\beta)) = [\![\mathsf{cadr}_k(\beta)]\!]^{\mathcal{A}}_\rho$, that contradicts the assumption. Hence we have $\alpha \equiv \beta$. Moreover, if $m \neq k$, we can define ρ' such that $\pi_m(\rho'(\alpha)) \neq \pi_k(\rho'(\alpha))$, and then similarly we have contradiction, and therefore, we have $m = k$. Hence we have $S_1 =_{\Lambda\mu_{\mathsf{cons}}} S_2$.

The logical relations on the stream models are given as follows.

Definition 6. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be stream models. A family of binary relations $\mathcal{R}^{(\cdot)}$ indexed with the expanded types is a *logical relation* on $\mathcal{A}_1$ and $\mathcal{A}_2$ if the following hold:

- $\mathcal{R}^\delta \subseteq \mathcal{A}_1^\delta \times \mathcal{A}_2^\delta$ and $\mathcal{R}^\sigma \subseteq \mathcal{A}_1^\sigma \times \mathcal{A}_2^\sigma$,
- $\mathcal{R}^X$ is non-empty for each X,
- $f_1 \mathcal{R}^{\sigma\to\delta} f_2$ if and only if $\forall s_1 \mathcal{R}^\sigma s_2.[f_1 \star s_1 \mathcal{R}^\delta f_2 \star s_2]$,
- $s_1 \mathcal{R}^\sigma s_2$ if and only if $\forall i \in \mathbb{N}.[\pi_i s_1 \mathcal{R}^{(\sigma)_i} \pi_i s_2]$.

Note that for any logical relation $\mathcal{R}$, we have that $s_1 \mathcal{R}^{\delta\times\sigma} s_2$ if and only if $\mathsf{Car}\, s_1 \mathcal{R}^\delta \mathsf{Car}\, s_2$ and $\mathsf{Cdr}\, s_1 \mathcal{R}^\sigma \mathsf{Cdr}\, s_2$. Therefore, if $a_1 \mathcal{R}^\delta a_2$ and $s_1 \mathcal{R}^\sigma s_2$, then we have $(a_1 :: s_1)\mathcal{R}^{\delta\times\sigma}(a_2 :: s_2)$.

For the logical relations, we will show the fundamental lemma in the following. The definition of the logical relations and the fundamental lemma can be straightforwardly generalized to n-ary logical relations.

Definition 7. Let $\mathcal{R}$ be a logical relation on stream models $\mathcal{A}_1$ and $\mathcal{A}_2$. For contexts Γ and Δ, we write $\rho_1 \mathcal{R}^{\Gamma;\Delta} \rho_2$ to denote the following:

$$\forall x \in \mathrm{dom}(\Gamma).[\rho_1(x)\mathcal{R}^{\Gamma(x)}\rho_2(x)], \text{ and } \forall \alpha \in \mathrm{dom}(\Delta).[\rho_1(\alpha)\mathcal{R}^{\Delta(\alpha)}\rho_2(\alpha)].$$

Proposition 3. *Let $\mathcal{R}$ be a logical relation on stream models $\mathcal{A}_1$ and $\mathcal{A}_2$. If $\Gamma;\Delta \vdash_{\Lambda\mu_{\mathsf{cons}}} t : \delta$ and $\rho_1 \mathcal{R}^{\Gamma;\Delta}\rho_2$, then we have $[\![t]\!]^{\mathcal{A}_1}_{\rho_1} \mathcal{R}^\delta [\![t]\!]^{\mathcal{A}_2}_{\rho_2}$.*

Proof. By induction on the derivation of $\Gamma;\Delta \vdash t : \delta$ and $\Gamma;\Delta \vdash S : \sigma$ simultaneously. In this proof, we omit the superscripts $\mathcal{A}_1$ and $\mathcal{A}_2$ for the meaning functions for simplicity.

Case (Ax) and (SAx). These cases are proved by the assumption $\rho_1 \mathcal{R}^{\Gamma;\Delta}\rho_2$.

Case (Lam) $\lambda x^{\delta'}.t^{\sigma\to\delta} : \delta' \times \sigma \to \delta$. Suppose that we have $s_1 \mathcal{R}^{\delta'\times\sigma} s_2$, that is, $\mathsf{Car} s_1 \mathcal{R}^{\delta'} \mathsf{Car} s_2$ and $\mathsf{Cdr} s_1 \mathcal{R}^\sigma \mathsf{Cdr} s_2$. Then we have $\rho[x \mapsto \mathsf{Car} s_1]\mathcal{R}^{\Gamma,x:\delta';\Delta}\rho[x \mapsto \mathsf{Car} s_2]$, so $[\![t]\!]_{\rho[x\mapsto \mathsf{Car} s_1]} \mathcal{R}^{\sigma\to\delta} [\![t]\!]_{\rho[x\mapsto \mathsf{Car} s_2]}$ holds by the induction hypothesis. Hence, we have $([\![t]\!]_{\rho[x\mapsto \mathsf{Car} s_1]} \star \mathsf{Cdr} s_1)\mathcal{R}^\delta([\![t]\!]_{\rho[x\mapsto \mathsf{Car} s_2]} \star \mathsf{Cdr} s_2)$, so we have $([\![\lambda x.t]\!]_{\rho_1} \star s_1)\mathcal{R}^\delta([\![\lambda x.t]\!]_{\rho_2} \star s_2)$ by the condition of $[\![\cdot]\!]$. This holds for any $s_1 \mathcal{R} s_2$, so we have $[\![\lambda x.t]\!]_{\rho_1} \mathcal{R}^{\delta'\times\sigma\to\delta} [\![\lambda x.t]\!]_{\rho_2}$.

Case (App) $t^{\delta'\times\sigma\to\delta} u^{\delta'} : \sigma \to \delta$. Suppose that we have $s_1 \mathcal{R}^\sigma s_2$. By the induction hypothesis, we have $[\![u]\!]_{\rho_1} \mathcal{R}^{\delta'} [\![u]\!]_{\rho_2}$, so we have $[\![u]\!]_{\rho_1} :: s_1 \mathcal{R}^{\delta'\times\sigma} [\![u]\!]_{\rho_2} :: s_2$. By the induction hypothesis for t, we have $([\![t]\!]_{\rho_1} \star ([\![u]\!]_{\rho_1} :: s_1))\mathcal{R}^\delta([\![t]\!]_{\rho_2} \star ([\![u]\!]_{\rho_2} :: s_2))$, which means $([\![tu]\!]_{\rho_1} \star s_1)\mathcal{R}^\delta([\![tu]\!]_{\rho_2} \star s_2)$ for any $s_1 \mathcal{R}^\sigma s_2$. Therefore we have $[\![tu]\!]_{\rho_1} \mathcal{R}^{\sigma\to\delta} [\![tu]\!]_{\rho_2}$.

Case (Mu) $\mu\alpha^\sigma.t^\delta : \sigma \to \delta$. Suppose that we have $s_1 \mathcal{R}^\sigma s_2$, and then we have $\rho_1[\alpha \mapsto s_1]\mathcal{R}^{\Gamma;\Delta,\alpha:\sigma}\rho_2[\alpha \mapsto s_2]$. By the induction hypothesis, we have $[\![t]\!]_{\rho_1[\alpha\mapsto s_1]} \mathcal{R}^\delta [\![t]\!]_{\rho_2[\alpha\mapsto s_2]}$, so we have $([\![\mu\alpha.t]\!]_{\rho_1} \star s_1)\mathcal{R}^\delta([\![\mu\alpha.t]\!]_{\rho_2} \star s_2)$ for any $s_1 \mathcal{R}^\sigma s_2$ by the condition of $[\![\cdot]\!]$. Therefore, we have $[\![\mu\alpha.t]\!]_{\rho_1} \mathcal{R}^{\sigma\to\delta} [\![\mu\alpha.t]\!]_{\rho_2}$.

Case (SApp) $t^{\sigma\to\delta}S^{\sigma}$. By the induction hypothesis, we have $[\![t]\!]_{\rho_1}\mathcal{R}^{\sigma\to\delta}[\![t]\!]_{\rho_2}$ and $[\![S]\!]_{\rho_1}\mathcal{R}^{\sigma}[\![S]\!]_{\rho_2}$, so we have $([\![t]\!]_{\rho_1}\star[\![S]\!]_{\rho_1})\mathcal{R}^{\delta}([\![t]\!]_{\rho_2}\star[\![S]\!]_{\rho_2})$, hence $[\![tS]\!]_{\rho_1}\mathcal{R}^{\delta}[\![tS]\!]_{\rho_2}$.

Case (Cons) $t^{\delta} :: S^{\sigma}$. By induction hypothesis, we have $[\![t]\!]_{\rho_1}\mathcal{R}^{\delta}[\![t]\!]_{\rho_2}$ and $[\![S]\!]_{\rho_1}\mathcal{R}^{\sigma}[\![S]\!]_{\rho_2}$, so we have $([\![t]\!]_{\rho_1} :: [\![S]\!]_{\rho_1})\mathcal{R}^{\delta\times\sigma}([\![t]\!]_{\rho_2} :: [\![S]\!]_{\rho_2})$, hence $[\![t :: S]\!]_{\rho_1}\mathcal{R}^{\delta\times\sigma}[\![t :: S]\!]_{\rho_2}$.

Case (Cdr) $\mathsf{cdr}S^{\delta\times\sigma}$. By induction hypothesis, we have $[\![S]\!]_{\rho_1}\mathcal{R}^{\delta\times\sigma}[\![S]\!]_{\rho_2}$, so we have $\mathsf{Cdr}[\![S]\!]_{\rho_1}\mathcal{R}^{\sigma}\mathsf{Cdr}[\![S]\!]_{\rho_2}$, hence $[\![\mathsf{cdr}S]\!]_{\rho_1}\mathcal{R}^{\sigma}[\![\mathsf{cdr}S]\!]_{\rho_2}$.

Corollary 1 (Fundamental lemma). *Let $\mathcal{R}$ be a logical relation on $\mathcal{A}_1$ and $\mathcal{A}_2$. If $;\vdash_{\Lambda\mu_{\mathsf{cons}}} t : \delta$ holds, then we have $[\![t]\!]^{\mathcal{A}_1}\mathcal{R}^{\delta}[\![t]\!]^{\mathcal{A}_2}$.*

Definition 8. Let $\mathcal{R}$ be a binary logical relation on $\mathcal{A}_1$ and $\mathcal{A}_2$. We call $\mathcal{R}$ *functional* if the following hold for any δ and σ:

$$\forall a_1 a_2 a_2'.[a_1\mathcal{R}^{\delta}a_2 \text{ and } a_1\mathcal{R}^{\delta}a_2' \Rightarrow a_2 = a_2'],$$
$$\forall s_1 s_2 s_2'.[s_1\mathcal{R}^{\sigma}s_2 \text{ and } s_1\mathcal{R}^{\sigma}s_2' \Rightarrow s_2 = s_2'].$$

A functional logical relation $\mathcal{R}$ is called *surjective* if $\forall a_2.\exists a_1.[a_1\mathcal{R}^{\delta}a_2]$ and $\forall s_2.\exists s_1.[s_1\mathcal{R}^{\sigma}s_2]$ holds for any δ and σ.

Lemma 2. *1. Suppose that a logical relation $\mathcal{R}$ on $\mathcal{A}_1$ and $\mathcal{A}_2$ is functional. For any closed typable t and u, if $[\![t]\!]^{\mathcal{A}_1} = [\![u]\!]^{\mathcal{A}_1}$ holds, then $[\![t]\!]^{\mathcal{A}_2} = [\![u]\!]^{\mathcal{A}_2}$ holds.*

2. Moreover, suppose that $\mathcal{R}$ is surjective. For any typable (open) terms t and u, if $[\![t]\!]^{\mathcal{A}_1}_{\rho_1} = [\![u]\!]^{\mathcal{A}_1}_{\rho_1}$ holds for any ρ_1, then $[\![t]\!]^{\mathcal{A}_2}_{\rho_2} = [\![u]\!]^{\mathcal{A}_2}_{\rho_2}$ holds for any ρ_2.

Proof. 1. By the fundamental lemma, we have $[\![t]\!]^{\mathcal{A}_1}\mathcal{R}[\![t]\!]^{\mathcal{A}_2}$ and $[\![u]\!]^{\mathcal{A}_1}\mathcal{R}[\![u]\!]^{\mathcal{A}_2}$. Hence we have $[\![t]\!]^{\mathcal{A}_2} = [\![u]\!]^{\mathcal{A}_2}$, since we have $[\![t]\!]^{\mathcal{A}_1} = [\![u]\!]^{\mathcal{A}_1}$ and $\mathcal{R}$ is functional.

2. For any $\rho_2 \in \mathsf{Env}^{\mathcal{A}_2}$ and any x, there exists a_1 such that $a_1\mathcal{R}\rho_2(x)$ since $\mathcal{R}$ is functional and surjective. Similarly, for any α, there exists s_1 such that $s_1\mathcal{R}\rho_2(\alpha)$. Then, we can define $\rho_1 \in \mathsf{Env}^{\mathcal{A}_1}$ as $\rho_1(x) = a_1$ for $a_1\mathcal{R}\rho_2(x)$ and $\rho_1(\alpha) = s_1$ for $s_1\mathcal{R}\rho_2(\alpha)$. We have $\rho_1\mathcal{R}\rho_2$, so we have $[\![t]\!]^{\mathcal{A}_1}_{\rho_1}\mathcal{R}[\![t]\!]^{\mathcal{A}_2}_{\rho_2}$ and $[\![u]\!]^{\mathcal{A}_1}_{\rho_1}\mathcal{R}[\![u]\!]^{\mathcal{A}_2}_{\rho_2}$ by Proposition 2. Since $[\![t]\!]^{\mathcal{A}_1}_{\rho_1} = [\![u]\!]^{\mathcal{A}_1}_{\rho_1}$ holds by the assumption, we have $[\![t]\!]^{\mathcal{A}_2}_{\rho_2} = [\![u]\!]^{\mathcal{A}_2}_{\rho_2}$ by functionality.

The following theorem gives a characterization of $=_{\Lambda\mu_{\mathsf{cons}}}$, corresponding to the well-known Friedman's theorem [7]: the extensional equality in $\lambda^{\to}$ is characterized by an arbitrary individual full type hierarchy with infinite bases.

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.*

Proof. The only-if part follows from the soundness.

For the if part, we will prove a more general result that for any (open) typable t and u, $t =_{\Lambda\mu_{\mathsf{cons}}} u$ holds if $[\![t]\!]^{\mathcal{F}}_{\rho} = [\![u]\!]^{\mathcal{F}}_{\rho}$ holds for any $\rho \in \mathsf{Env}^{\mathcal{F}}$. Consider the

open term model $\mathcal{T}$, and then we have that $t =_{\Lambda\mu_{\mathsf{cons}}} u$ holds if and only if $[\![t]\!]^{\mathcal{T}}_{\rho} = [\![u]\!]^{\mathcal{T}}_{\rho}$ holds for any ρ. Hence, it is sufficient to prove that there exists a surjective functional logical relation $\mathcal{R}$ from $\mathcal{F}$ to $\mathcal{T}$. This is proved by induction on types.

Case X. $\mathcal{T}^X$ is countably infinite set, and $\mathcal{F}^X$ is infinite by the assumption, so there is a surjective function from $\mathcal{F}^X$ to $\mathcal{T}^X$.

Case $\sigma \to \delta$. (Functionality) Suppose $f\mathcal{R}^{\sigma\to\delta}f'$ and $f\mathcal{R}^{\sigma\to\delta}f''$ for $f \in \mathcal{F}^{\sigma\to\delta}$ and $f', f'' \in \mathcal{T}^{\sigma\to\delta}$. By the definition of the logical relation, for any $s\mathcal{R}^{\sigma}s'$, we have $f(s)\mathcal{R}^{\delta}f'\star s'$ and $f(s)\mathcal{R}^{\delta}f''\star s'$. By the induction hypothesis, $\mathcal{R}^{\delta}$ is functional, and hence $f'\star s' = f''\star s'$. By the induction hypothesis again, $\mathcal{R}^{\sigma}$ is surjective, so, for any $s' \in \mathcal{T}^{\sigma}$, we have some $s \in \mathcal{F}^{\sigma}$ such that $s\mathcal{R}^{\sigma}s'$. Therefore, we have $f'\star s' = f''\star s'$ for any s', so $f' = f''$ by the extensionality.

(Surjectivity) Let $f' \in \mathcal{T}^{\sigma\to\delta}$. If $s\mathcal{R}^{\sigma}s'$, there exists $a \in \mathcal{F}^{\delta}$ such that $a\mathcal{R}^{\delta}f'\star s'$ since $\mathcal{R}^{\delta}$ is surjective by the induction hypothesis. Fix such an a for each s', and define $f \in \mathcal{F}^{\sigma} \to \mathcal{F}^{\delta} = \mathcal{F}^{\sigma\to\delta}$ as follows:

$$f(s) := \begin{cases} a & (s\mathcal{R}^{\sigma}s' \text{ and } a\mathcal{R}^{\delta}f'\star s' \text{ for some } s') \\ a_0 & (\text{otherwise}), \end{cases}$$

where a_0 is a fixed element in $\mathcal{F}^{\delta}$. What we have to show is $f\mathcal{R}^{\sigma\to\delta}f'$, that is, for any $s\mathcal{R}^{\sigma}s'$, $f(s)\mathcal{R}^{\delta}f'\star s'$, and it holds by the definition of f.

Case $\Pi_{i\in\mathbb{N}}\delta_i$. (Functionality) Suppose $s\mathcal{R}^{\Pi_i\delta_i}s'_1$ and $s\mathcal{R}^{\Pi_i\delta_i}s'_2$. We have $\pi_i s\mathcal{R}^{\delta_i}\pi_i s'_1$ and $\pi_i s\mathcal{R}^{\delta_i}\pi_i s'_2$, so $\pi_i s'_1 = \pi_i s'_2$ for any i by the functionality of $\mathcal{R}^{\delta_i}$. Let $s'_1 = [S_1]$ and $s'_2 = [S_2]$, and then we have $\mathsf{cadr}_i(S_1) =_{\Lambda\mu_{\mathsf{cons}}} \mathsf{cadr}_i(S_2)$ for any i. By Lemma 1, we have $S_1 =_{\Lambda\mu_{\mathsf{cons}}} S_2$, that is, $s'_1 = s'_2$ holds.

(Surjectivity) Let $s' \in \mathcal{T}^{\Pi_{i\in\mathbb{N}}\delta_i}$. By the surjectivity of $\mathcal{R}^{\delta_i}$, there exists $a_i \in \mathcal{F}^{\delta_i}$ such that $a_i\mathcal{R}^{\delta_i}\pi_i s'$ for each i. Then, for $s = \langle a_0, a_1, \cdots\rangle \in \Pi_{i\in\mathbb{N}}\mathcal{A}^{\delta_i} = \mathcal{A}^{\Pi_{i\in\mathbb{N}}\delta_i}$, we have $\pi_i s = a_i\mathcal{R}^{\delta_i}\pi_i s'$ for any i, so we have $s\mathcal{R}^{\Pi_{i\in\mathbb{N}}\delta_i}s'$.

4 Relationship with Existing Systems

In this section, we discuss the relationship between our systems and the existing related type systems such as Pagani and Saurin's $\Lambda_{\mathcal{S}}$ in [12, 16], Parigot's original $\lambda\mu$-calculus in [13], and Gaboardi and Saurin's $\Lambda_{\mathcal{S}}$ in [8].

4.1 Pagani and Saurin's $\Lambda_{\mathcal{S}}$ and Parigot's $\lambda\mu$

Compared to Pagani and Saurin's $\Lambda_{\mathcal{S}}$, we restrict functional types to ones from streams to terms, but, regarding typability, $\Lambda\mu_{\mathsf{cons}}$ is not less than $\Lambda_{\mathcal{S}}$. Indeed, we can give a translation from [12, 16] to $\Lambda\mu_{\mathsf{cons}}$ preserving typability.

The types of $\Lambda_{\mathcal{S}}$ are defined as

$$\mathcal{T} ::= o \mid \mathcal{T} \to \mathcal{T} \mid \mathcal{S} \Rightarrow \mathcal{T} \qquad\qquad \mathcal{S} ::= \bot \mid \mathcal{T} \to \mathcal{S},$$

and $\mathcal{T}$ and $\mathcal{S}$ correspond to term types and stream types in $\Lambda\mu_{\mathsf{cons}}$, respectively. We consider types modulo the equivalence defined from the axiom $(\mathcal{T} \to \mathcal{S}) \Rightarrow$

$\mathcal{T}' \equiv \mathcal{T} \to (\mathcal{S} \Rightarrow \mathcal{T}')$. The stream type of the form $\mathcal{T} \to \mathcal{S}$ corresponds to the product type $\mathcal{T} \times \mathcal{S}$ in our system. Our notation seems natural to represent types of streams constructed from the pair of head and tail of the streams, whereas the notation of $\Lambda\mu_{\mathsf{cons}}$ also seems natural under correspondence between streams and (evaluation) contexts. For example, a stream constructed from a head t and a tail α is represented as a context $[]t\alpha$ in $\Lambda\mu$, the type of which should be the type of the function expected to fill the hole. Therefore, when t has a type $\mathcal{T}$ and α has a type $\mathcal{S}$, the context should have the type $\mathcal{T} \to \mathcal{S}$ as [5].

A formal definition of the translation from $\Lambda\mu_{\mathcal{S}}$ can be given as follows.

Definition 9. *Fix a type variable O in $\Lambda\mu_{\mathsf{cons}}$. We define the translation $\overline{(\cdot)}$ from the $\Lambda_{\mathcal{S}}$-types to the $\Lambda\mu_{\mathsf{cons}}$-types by*

$$\overline{o} = [O] \to O \qquad \overline{\mathcal{T} \to \mathcal{T}'} = \overline{\mathcal{T}} \times \overline{\mathcal{T}'}_s \to \overline{\mathcal{T}'}_t \qquad \overline{\mathcal{S} \Rightarrow \mathcal{T}} = \overline{\mathcal{S}} \to \overline{T}$$
$$\overline{\bot} = [O] \qquad \overline{\mathcal{T} \to \mathcal{S}} = \overline{\mathcal{T}} \times \overline{\mathcal{S}},$$

where $\overline{\mathcal{T}}$ is always an arrow type and we put $\overline{\mathcal{T}} = \overline{\mathcal{T}}_s \to \overline{\mathcal{T}}_t$.

Then, we can easily show that $\mathcal{T} \equiv \mathcal{T}'$ implies $\overline{\mathcal{T}} \sim \overline{\mathcal{T}'}$, and the following is proved by a straightforward induction.

Proposition 4. *If $\Gamma \vdash_{\Lambda_{\mathcal{S}}} t : \mathcal{T}; \Delta$, then $\overline{\Gamma}; \overline{\Delta} \vdash_{\Lambda\mu_{\mathsf{cons}}} t : \overline{\mathcal{T}}$.*

Example 2. Pagani and Saurin showed in [12] that the control operator `call/cc` represented as the $\Lambda\mu$-term $\lambda x.\mu\alpha.x(\lambda y.\mu\beta.y\alpha)\alpha$ has a type

$$\mathcal{T}_{\texttt{call/cc}} = (((\mathcal{S}_\alpha \Rightarrow \mathcal{T}) \to (\mathcal{S}_\beta \Rightarrow \mathcal{T})) \to (\mathcal{S}_\alpha \Rightarrow \mathcal{T}')) \to (\mathcal{S}_\alpha \Rightarrow \mathcal{T}')$$

for any term types $\mathcal{T}$, $\mathcal{T}'$, and any stream types $\mathcal{S}_\alpha$, $\mathcal{S}_\beta$ in their type system. Hence, in $\Lambda\mu_{\mathsf{cons}}$, `call/cc` has the type

$$\overline{\mathcal{T}_{\texttt{call/cc}}} = ((\sigma_\alpha \to X) \times \sigma_\beta \to X) \times \sigma_\alpha \to Y) \times \sigma_\alpha \to Y.$$

This example suggests a relationship between (the first-order fragment of) the original typed $\lambda\mu$-calculus and our type system. The type of `call/cc` in the original system is known as Peirce's law $((A \to B) \to A) \to A$, which shows the $\lambda\mu$-calculus corresponds to the classical logic. On the other hand, our system seems intuitionistic whereas `call/cc` has a type in it. That can be understood by the fact that the type of `call/cc` in our system is a negative translation of the Peirce's law.

Saurin [16] and van Bakel et al. [19] showed that their type systems correspond to the image of negative translations from the classical logic to the intuitionistic logic, and we can show the same result for our system as follows. The following is a simplified variant of the composition of the translation from $\lambda\mu$ to $\Lambda_{\mathcal{S}}$ in [16] and the translation in Definition 9. It corresponds to the continuation-passing-style translation of Thielecke [18].

Definition 10 (Negative translation). *We fix a type variable O and an arbitrary stream type θ (for example, we take $\theta = [O]$). We write $\neg\sigma$ for $\sigma \to O$. The negative translations $\overline{(\cdot)}$ from the implicational formulas to term types in $\Lambda\mu_{\mathsf{cons}}$ and $(\cdot)^\bullet$ from the implicational formulas to stream 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 5. *If $\Gamma \vdash t : A; \Delta$ holds in the Parigot's original typed $\lambda\mu$-calculus in [13], then $\neg\Gamma^\bullet; \Delta^\bullet \vdash t : \neg A^\bullet$ holds in $\Lambda\mu_{\mathsf{cons}}$.*

As a corollary of the proposition, every typable λ-term in the simply-typed λ-calculus, denoted by $\lambda^\to$, is typable in $\Lambda\mu_{\mathsf{cons}}$. Moreover, for the λ-terms, we have a more direct translation on types, which is similar to the translation given in [19]. Fix an arbitrary type variable O, and define the translation on the simple types as

$$(\theta_0 \to \cdots \to \theta_n \to X)^\# = \theta_0^\# \times \cdots \times \theta_n^\# \times [O] \to X.$$

Then, for any λ-term t, if $\Gamma \vdash t : \theta$ holds in $\lambda^\to$, then we have $\Gamma^\#; \vdash t : \theta^\#$ in $\Lambda\mu_{\mathsf{cons}}$.

Hence, regarding typability, $\Lambda\mu_{\mathsf{cons}}$ can be considered to include $\lambda^\to$, $\lambda\mu$, and $\Lambda\mu_{\mathcal{S}}$. On the other hand, $\Lambda\mu_{\mathsf{cons}}$ can type properly wider class of terms than the other systems due to the recursive types, that is, $\Lambda\mu_{\mathsf{cons}}$ is not conservative over $\lambda^\to$ with respect to typability. An example is the following. Let $\delta = [X] \to X$ and $\Gamma = f : \delta, a : X$. Then, both $\Gamma; \vdash fa : \delta$ and $\Gamma; \vdash faa : \delta$ hold since we have $[X] \sim X \times [X]$. Therefore, the term $w(x(fa))(x(faa))$ has the type δ under the context $\Gamma, w : \delta \times \delta \times [X] \to X, x : \delta \times [X] \to X$ in $\Lambda\mu_{\mathsf{cons}}$. The typability of this term seems natural, when we think f as a function on streams of the type $[X]$, and applications fa and faa as partial application of f, and then both fa and faa are also functions on streams of the type $[X]$. On the other hand, in the other systems, the types of yz and yzz must be distinct, and therefore $w(x(yz))(x(yzz))$ has no type. Note that the term in this example is a λ-term, and hence it shows the typability in $\Lambda\mu_{\mathsf{cons}}$ is not conservative over $\lambda^\to$.

Proposition 6. *Every typable term in either $\lambda^\to$, $\lambda\mu$, or $\Lambda_{\mathcal{S}}$ is typable in $\Lambda\mu_{\mathsf{cons}}$. On the other hand, there is a λ-term which is typable in $\Lambda\mu_{\mathsf{cons}}$, and not typable in neither $\lambda^\to$, $\lambda\mu$, nor $\Lambda_{\mathcal{S}}$.*

4.2 Gaboardi and Saurin's $\Lambda_{\mathcal{S}}$ and Recursive Types

Gaboardi and Saurin [8] proposed the coercion operator from stream to terms, which enables to represent functions returning streams such as cdr. The proposed system $\Lambda_{\mathcal{S}}$ is defined based on $\Lambda\mu_{\mathcal{S}}$, and contains a coercion operator from streams to terms and its inverse, which are denoted by tm and st in this paper, with typing rules

$$\frac{S : \sigma}{\mathsf{tm}(S) : \mathsf{Str}(\sigma)} \qquad \frac{t : \mathsf{Str}(\sigma)}{\mathsf{st}(t) : \sigma},$$

where $\mathsf{Str}(\sigma)$ is the term type for the coerced streams of type σ. The stream models can be extended to the coercion operators in a straightforward way by defining $\mathcal{A}^{\mathsf{Str}(\sigma)} = \mathcal{A}^{\sigma}$, $[\![\mathsf{tm}(S)]\!] = [\![S]\!]$, and $[\![\mathsf{st}(t)]\!] = [\![t]\!]$.

Gaboardi and Saurin's $\Lambda_{\mathcal{S}}$ also considers recursive types. We adopt the restricted form of the recursive types for streams. We can define a translation from $\Lambda\mu_{\mathsf{cons}}$ to $\Lambda_{\mathcal{S}}$ preserving typability. See Appendix B.

The recursive stream types are essential if we consider recursive functions in typed setting. For example, consider the nth function. Let Y be a fixed point combinator in the untyped λ-calculus. The function nth is defined in the untyped $\Lambda\mu_{\mathsf{cons}}$ 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 function nth can be typed with help of the recursive structure of stream types under the assumption of existence of typed fixed-point operators.

Suppose the unit type 1 and the type N of the natural numbers as the base term types, the only constant $[()]$ of the type $[1]$, and some primitive operations on N such as $\mathsf{pred} : N \times [1] \to N$ and $\mathsf{ifzero} : N \times \delta \times \delta \times [1] \to \delta$. We can consider the map $t : \delta \mapsto t :: [()] : \delta \times [1]$ as a coercion from terms to streams, and we use the following notations.

$$\underline{\delta} = t \times [1] \qquad \underline{t} = t :: [()] \qquad \mu\underline{x^{\tau}}.t = \lambda x^{\tau}.\mu\alpha^{[1]}.t,$$

where α is a fresh variable, and we have $(\mu\underline{x}.t)\underline{u} = t[x := u]$. Let fix_{δ} be the typed fixed-point operator, the type of which is $\underline{(\delta \to \delta)} \to \delta$, and suppose that $\mathsf{fix}_{\delta}(\underline{F}) = F(\mathsf{fix}_{\delta}(\underline{F}))$ for any $F : \underline{\delta} \to \delta$. Note that an untyped term satisfying this equation is $\mu\underline{f}.(\lambda x.f(xx))(\lambda x.f(xx))$. Then, we can define the typed version of the nth function as $\mathsf{fix}_{\delta}(\underline{F})$, where $\delta = [\delta_0] \to \underline{N} \to N$ and

$$F = \mu\underline{f^{\delta}}.\mu\alpha^{[\delta_0]}.\mu\underline{m^{N}}.\mathsf{ifzero}\,m\,(\mathsf{car}\,\alpha)\,(f(\mathsf{cdr}\,\alpha)(\underline{\mathsf{pred}\,m}))\,[()].$$

Note that, by the recursive structure of the type $[\delta_0]$, both α and $\mathsf{cdr}\,\alpha$ have the same type $[\delta_0]$, and F indeed has the type $\underline{\delta} \to \delta$, and we have $\mathsf{nth} = F(\underline{\mathsf{nth}})$, and hence $\mathsf{nth}\,S\,\underline{0} = \mathsf{car}\,S$ and $\mathsf{nth}\,S\,\underline{m+1} = \mathsf{nth}\,(\mathsf{cdr}\,S)\,\underline{m}$ hold for any stream S of the type $[\delta_0]$ and any m of the type N as we expect.

Definition 11 (Extended Stream Models). *We call $\mathcal{A}$ is an extended stream model if $\mathcal{A}$ is a stream model for the typed $\Lambda\mu_{\mathsf{cons}}$ with the base term types 1 and N and the constant $[()]$ of the type $[1]$ satisfying the following conditions.*

- *A fixed element $\bullet$ is in $\mathcal{A}^1$, and $\pi_n([\![[()]]\!]^{\mathcal{A}}) = \bullet$ for any $n \in \mathbb{N}$. We denote $\bullet$ for $[\![[()]]\!]^{\mathcal{A}}$, and, for $a \in \mathcal{A}^{\delta}$, we denote $\underline{a}$ for $a :: \bullet$ which is an element of $\mathcal{A}^{\underline{\delta}}$, and then we have $[\![\underline{t}]\!]^{\mathcal{A}} = \underline{[\![t]\!]}^{\mathcal{A}}$.*

– $\mathcal{A}^N$ *includes* $\mathbb{N}$*, and the primitive operators on* N *are interpreted in an appropriate way such as* $[\![\mathsf{pred}]\!]^{\mathcal{A}}$ *is an element of* $\mathcal{A}^{\overline{N}\to N}$ *such that* $[\![\mathsf{pred}]\!]^{\mathcal{A}}\star\underline{n+1} = n$*.*
– $\mathsf{Fix}_\delta := [\![\mathsf{fix}_\delta]\!]^{\mathcal{A}}$ *is an element of* $\mathcal{A}^{\underline{(\delta\to\delta)\to\delta}}$ *such that, for any* $f \in \mathcal{A}^{\underline{\delta\to\delta}}$*,*

$$\mathsf{Fix}_\delta\star\underline{f} = f\star(\underline{\mathsf{Fix}_\delta\star\underline{f}}).$$

Example 3. An example of the extended stream model can be constructed as the pointed CPO model.

– $\mathcal{D}^1$ and $\mathcal{D}^N$ are the flat domains constructed from $\{\bullet\}$ and $\mathbb{N}$, respectively.
– $\mathcal{D}^{\sigma\to\delta} := [\mathcal{D}^\sigma \to \mathcal{D}^\delta]$ is the set of continuous functions, and $\mathcal{D}^\sigma := \Pi_{i\in\mathbb{N}}\mathcal{D}^{(\sigma)_i}$.
– For any $p \in \mathcal{D}^{\underline{\delta\to\delta}} = \mathcal{D}^{\underline{\delta}\to\delta} \times \mathcal{D}^{[1]}$,

$$\mathsf{Fix}_\delta(p) := \bigsqcup_{n\in\mathbb{N}} (\overline{\lambda}d \in \mathcal{D}^\delta.(\mathsf{Car}\,p)(d :: (\mathsf{Cdr}\,p)))^n(\bot).$$

Note that, in particular, when $p = \underline{f}$ for $f \in \mathcal{A}^{\underline{\delta}\to\delta}$, we have

$$\mathsf{Fix}_\delta(\underline{f}) = \bigsqcup_{n\in\mathbb{N}} (\overline{\lambda}d \in \mathcal{D}^\delta.f(\underline{d}))^n(\bot).$$

Recursive functions returning streams are defined with the help of the coercion $\mathsf{tm}(S)$. An example is the function alt interleaving two streams as

$$\mathsf{alt}\,(t_1 :: t_2 \cdots)\,(u_1 :: u_2 \cdots) = t_1 :: u_1 :: t_2 :: u_2 \cdots,$$

which is simply defined as $\mathsf{alt} = \mathsf{fix}_\delta\,\underline{G}$ with $\delta = [\tau] \to [\tau] \to \mathsf{Str}([\tau])$ and

$$G = \mu\underline{g^\delta}.\mu\alpha^{[\tau]}.\mu\beta^{[\tau]}.\mathsf{tm}(\mathsf{car}\alpha :: \mathsf{car}\beta :: \mathsf{st}(g(\mathsf{cdr}\alpha)(\mathsf{cdr}\beta))),$$

and then alt has the type δ.

Note that we have to use the extra coercion $(\underline{\cdot})$ over st of $\Lambda_{\mathcal{S}}$, because $\mathsf{st}(t)$ is typed only when t is a coerced stream.

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. P.-L. Curien and H. Herbelin. The duality of computation. In *5th ACM SIGPLAN International Conference on Functional Programming (ICFP'00)*, pages 233–243, 2000.
6. U. de'Liguoro. The approximation theorem for the $\Lambda\mu$-calculus. *Mathematical Structures in Computer Science*, to appear.
7. H. Friedman. Equality between functionals. In Parikh, R., editor, *Logic Colloquium*, volume 453 of *Lecture Notes in Mathematics*, pages 22–37. Springer, 1973.
8. M. Gaboardi and A. Saurin. A foundational calculus for computing with streams. In *12th Italian Conference on Theoretical Computer Science*, 2010.
9. R. Milner. A complete inference system for a class of regular begaviours. *Journal of Computer and System Sciences*, 28:439–466, 1984.
10. 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.
11. K. Nakazawa and T. Nagai. Reduction system for extensional lambda-mu calculus. unpublished manuscript, 2014. Available at `http://www.fos.kuis.kyoto-u.ac.jp/~knak/papers/manuscript/LmC-red.pdf`.
12. M. Pagani and A Saurin. Stream associative nets and $\lambda\mu$-calculus. Research Report 6431, INRIA, 2008.
13. 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.
14. A. Salomaa. Two complete axiom systems for the algebra of regular events. *Journal of the Associations for Computing Machinery*, 13(1):158–169, 1966.
15. 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.
16. A. Saurin. Typing streams in the $\Lambda\mu$-calculus. *ACM Transactions on Computational Logic*, 11, 2010.
17. T. Streicher and B. Reus. Classical logic, continuation semantics and abstract machines. *Journal of Functional Programming*, 8(6):543–572, November 1998.
18. H. Thielecke. *Categorical structure of continuation passing style*. PhD thesis, University of Edinburgh, 1997.
19. 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 theorem. 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 $0 \leq i \leq |\boldsymbol{\delta}| - 1$.

Theorem 4. *For any types δ and δ', $\delta \sim \delta'$ is equivalent to $\langle\!|\delta|\!\rangle = \langle\!|\delta'|\!\rangle$.*

We divide the theorem into the following two propositions.

Proposition 7. *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 3. *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 12. *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 4. *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

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

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

Proposition 8. *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 4, $d(\delta_i)$ 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 3, 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 Embedding into Gaboardi and Saurin's $\Lambda_{\mathcal{S}}$

Definition 13. *We define a translation from $\Lambda\mu_{\mathsf{cons}}$ to $\Lambda_{\mathcal{S}}$ as follows.*

On types:

$$\underline{X} = X \qquad\qquad \underline{\sigma \to \delta} = \underline{\sigma} \Rightarrow \underline{\delta}$$
$$\underline{[\delta_0, \cdots, \delta_{n-1}]} = \mu\mathcal{X}.\delta_0 \to \cdots \to \delta_{n-1} \to \mathcal{X} \qquad \underline{\delta \times \sigma} = \underline{\delta} \to \underline{\sigma}$$

On terms:

$$\underline{x} = x \qquad\qquad \underline{\alpha} = \alpha$$
$$\underline{\lambda x.t} = \lambda x.\underline{t} \qquad\qquad \underline{t :: S} = [\underline{t}; \underline{S}]$$
$$\underline{tu} = \underline{t}\,\underline{u} \qquad\qquad \underline{\mathsf{cdr}S} = \sigma((\lambda x.\mu\alpha.\tau(\alpha))\underline{S})$$
$$\underline{\mu\alpha.t} = \mu\alpha.\underline{t}$$
$$\underline{tS} = \underline{t}\,\underline{S}$$

Proposition 9. *1. If $\Gamma; \Delta \vdash_{\Lambda\mu_{\mathsf{cons}}} t : \delta$, then $\underline{\Gamma} \vdash_{\Lambda_{\mathcal{S}}} \underline{t} : \underline{\delta}; \underline{\Delta}$.*

2. If $t = u$ is derivable in $\Lambda\mu_{\mathsf{cons}}$ without use of (surj), then $\underline{t} = \underline{u}$ in $\Lambda_{\mathcal{S}}$.