

# Extensional Models of Typed Lambda-mu Calculus

Koji Nakazawa

Graduate School of Informatics, Kyoto University

knak@kuis.kyoto-u.ac.jp

This paper shows that the stream models introduced by Nakazawa and Katsumata can be adapted to a typed setting. A variant of the type system of van Bakel, Barbanera, and de'Liguoro for an extension of the  $\Lambda\mu$ -calculus, called  $\Lambda\mu_{\text{cons}}$ , is sound and complete with respect to the stream models. It is proved that any individual stream model with whole function spaces and infinite domains for base types characterizes the extensional equality of  $\Lambda\mu_{\text{cons}}$ . This result corresponds to Friedman's theorem for the simply-typed  $\lambda$ -calculus and the full type hierarchies.

## 1 Introduction

Parigot's  $\lambda\mu$ -calculus was originally introduced as a term assignment system for the classical natural deduction in [12], and a lot of typed and untyped variants have been widely studied from both sides of logic and computer science. Regarding untyped variants, Saurin showed that a slight extension, called the  $\Lambda\mu$ -calculus, enjoys the separation theorem, which does not hold for the original  $\lambda\mu$ -calculus as shown by David and Py [4].

For the untyped  $\Lambda\mu$ -calculus, Nakazawa and Katsumata [9] gave a class of extensional models, called *stream models*, which consist of two domains,  $D$  of individual data and  $S$  of streams, satisfying the domain equations:  $S = D \times S$  and  $D = S \rightarrow D$ . It can be seen as a restriction of Streicher and Reus's continuation semantics [16] by identifying  $D$  and the domain of "results". It is shown in [10] that the untyped  $\Lambda\mu$ -calculus (and its extension  $\Lambda\mu_{\text{cons}}$ ) is sound and complete for the stream models.

In this paper, the stream models are adapted to a typed setting. We consider a type system for the calculus  $\Lambda\mu_{\text{cons}}$  in [10], which is based on the type systems introduced in [18, 19, 5] and extended by restricted forms of recursive types for streams. The results of this paper are: (1) we introduce stream models for the typed  $\Lambda\mu_{\text{cons}}$ , (2) we prove soundness and completeness, and (3) we prove that the extensional equality of the typed  $\Lambda\mu_{\text{cons}}$  is characterized by any individual stream model with whole function spaces and infinite domains for base types. The third result is proved by means of the fundamental lemma for the logical relations on the stream models, and corresponds to Friedman's theorem for the simply typed  $\lambda$ -calculus and the full type hierarchies. Furthermore, we discuss an extension of the calculus and the stream models with typed fixed-point operators.

## 2 Type System for $\Lambda\mu_{\text{cons}}$

First, we introduce the typed variant of the calculus  $\Lambda\mu_{\text{cons}}$  in [10].

The calculus  $\Lambda\mu_{\text{cons}}$  is a conservative extension of the  $\Lambda\mu$ -calculus [14] with explicit stream syntax. In [10], we introduced the untyped  $\Lambda\mu_{\text{cons}}$  in order to consider a reduction system and to construct the open term model as a stream model. We will straightforwardly construct an open term model from the typed  $\Lambda\mu_{\text{cons}}$  as a stream model in the following sections, and the open term model plays a main role in the proof of Friedman's theorem. The calculus  $\Lambda\mu_{\text{cons}}$  is also close to the  $\lambda\mu$ -calculus considered in [16], which explicitly has the expressions for continuations but no *cdr* operator.

**Definition 2.1** (Terms and streams) We are supposed to have two sorts of variables: term variables, denoted by  $x, y, z, \dots$ , and stream variables, denoted by  $\alpha, \beta, \dots$ . The set of the term variables and the stream variables are denoted by TV and SV, respectively.

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

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

The set of the  $\Lambda\mu_{\text{cons}}$ -terms and the set of  $\Lambda\mu_{\text{cons}}$ -streams are denoted by Tm and St, respectively. Occurrences of  $x$  in  $\lambda x. t$  and occurrences of  $\alpha$  in  $\mu \alpha. t$  are considered to be bound. A variable occurrence which is not bound is called *free*.

The type assignment system for  $\Lambda\mu_{\text{cons}}$  given here is inspired by the type systems in [18, 19] for the  $\lambda\mu$ -calculus and in [5] for the  $\Lambda\mu$ -calculus, and adopting recursive types to represent types of streams similarly to [7]. 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 recursive types and is just an abbreviation for  $\mu \chi. \delta_0 \times \delta_1 \times \chi$ . Hence, the stream type  $[\delta_0, \delta_1]$  intuitively represents the infinite product type  $\delta_0 \times \delta_1 \times \delta_0 \times \delta_1 \times \dots$ . In the following sections, the semantical discussion will be done for such infinite product types, called the expanded types in this paper, and we adopt the expression such as  $[\delta_0, \delta_1]$  for stream types only for the syntactic finiteness of the type system. 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 [13, 8, 1, 2].

**Definition 2.2** (Type assignment for  $\Lambda\mu_{\text{cons}}$ ) The types consist of two sorts, *term types* and *stream types*, which are inductively defined as

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

where  $X$  ranges over base types, and  $[\delta_0, \dots, \delta_{n-1}]$  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.

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

A term context  $\Gamma$  and a stream context  $\Delta$  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  $\Lambda\mu_{\text{cons}}$  are the following.

$$\begin{array}{c} \frac{}{\Gamma, x : \delta \mid \Delta \vdash x : \delta} \text{ (Ax)} \qquad \frac{}{\Gamma \mid \Delta, \alpha : \sigma \vdash \alpha : \sigma} \text{ (SAx)} \\[10pt] \frac{\Gamma, x : \delta \mid \Delta \vdash t : \sigma \rightarrow \delta'}{\Gamma \mid \Delta \vdash \lambda x. t : \delta \times \sigma \rightarrow \delta'} \text{ (Lam)} \qquad \frac{\Gamma \mid \Delta \vdash t : \delta \times \sigma \rightarrow \delta' \quad \Gamma \mid \Delta \vdash u : \delta}{\Gamma \mid \Delta \vdash tu : \sigma \rightarrow \delta'} \text{ (App)} \\[10pt] \frac{\Gamma \mid \Delta, \alpha : \sigma \vdash t : \delta}{\Gamma \mid \Delta \vdash \mu \alpha. t : \sigma \rightarrow \delta} \text{ (Mu)} \qquad \frac{\Gamma \mid \Delta \vdash t : \sigma \rightarrow \delta \quad \Gamma \mid \Delta \vdash S : \sigma}{\Gamma \mid \Delta \vdash tS : \delta} \text{ (SApp)} \\[10pt] \frac{\Gamma \mid \Delta \vdash t : \delta \quad \Gamma \mid \Delta \vdash S : \sigma}{\Gamma \mid \Delta \vdash t :: S : \delta \times \sigma} \text{ (Cons)} \\[10pt] \frac{\Gamma \mid \Delta \vdash S : \delta \times \sigma}{\Gamma \mid \Delta \vdash \text{car } S : \delta} \text{ (Car)} \qquad \frac{\Gamma \mid \Delta \vdash S : \delta \times \sigma}{\Gamma \mid \Delta \vdash \text{cdr } S : \sigma} \text{ (Cdr)} \end{array}$$

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

In the following, we suppose that each term (stream) variable is assigned a fixed term (stream, respectively) type, that is, we adopt the so-called Church-style. The variable  $x$  assigned the type  $\delta$  is also denoted by  $x^\delta$ . For terms and streams, we also use the annotation such as  $t^\delta$  and  $S^\sigma$  to make their types explicit.

**Definition 2.3** (Extensional equality) The axiom schema of  $\Lambda\mu_{\text{cons}}$  are the following:

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

where  $t$  contains no free  $x$  in  $(\eta_T)$  and  $t$  contains no free  $\alpha$  in  $(\eta_S)$ . The reflexive, symmetric, transitive, and compatible relation  $=_{\Lambda\mu_{\text{cons}}}$  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  $t_1 =_{\Lambda\mu_{\text{cons}}} t_2$  hold. For simplicity, we write just  $t_1 =_{\Lambda\mu_{\text{cons}}} t_2$  to mean  $\Gamma \mid \Delta \vdash t_1 = t_2 : \delta$  for some  $\Gamma, \Delta$ , and  $\delta$ .

We use the following abbreviations

$$\text{cdr}^0 S \equiv S \quad \text{cdr}^{i+1} S \equiv \text{cdr}(\text{cdr}^i S) \quad \text{cadr}^i(S) \equiv \text{car}(\text{cdr}^i S)$$

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

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

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

Intuitively, each stream type represents a infinite product of term types, which is defined as the expanded types as follows.

**Definition 2.4** 1. The *expanded types* are defined by

$$\delta ::= X \mid \sigma \rightarrow \delta \quad \sigma ::= \prod_{i \in \mathbb{N}} \delta_i$$

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

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

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

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

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

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

$$\langle X \rangle = X \quad \langle \sigma \rangle = \prod_{i \in \mathbb{N}} \langle (\sigma)_i \rangle \quad \langle \sigma \rightarrow \delta \rangle = \langle \sigma \rangle \rightarrow \langle \delta \rangle$$

Note that the relation  $\sqsubset$  defined as

$$\sigma \sqsubset \sigma \rightarrow \delta \quad \delta \sqsubset \sigma \rightarrow \delta \quad \delta_i \sqsubset \prod_{i \in \mathbb{N}} \delta_i$$

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

The following is shown in [10].

**Proposition 2.5**  $\delta \sim \delta' \ (\sigma \sim \sigma')$  iff  $\langle \delta \rangle = \langle \delta' \rangle \ (\langle \sigma \rangle = \langle \sigma' \rangle)$ , respectively).

As we will discuss in Section 6, the types of  $\Lambda\mu_{\text{cons}}$  can be seen as a restriction of the types of  $\Lambda_{\mathcal{J}}$  of [11], but, regarding typability,  $\Lambda\mu_{\text{cons}}$  is not less than  $\Lambda_{\mathcal{J}}$ . Furthermore, it has some fundamental properties such as subject reduction and strong normalization for a reduction system for  $\Lambda\mu_{\text{cons}}$  [10].

### 3 Stream Models for Typed $\Lambda\mu_{\text{cons}}$

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

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

**Definition 3.1** 1. A *stream pre-model*  $\mathcal{A}$  for the typed  $\Lambda\mu_{\text{cons}}$  consists of the following:

- $\{\mathcal{A}^\delta\}_\delta$  and  $\{\mathcal{A}^\sigma\}_\sigma$ : families of sets indexed by the expanded types.
- A function  $(\star_{\sigma, \delta}) : \mathcal{A}^{\sigma \rightarrow \delta} \times \mathcal{A}^\sigma \rightarrow \mathcal{A}^\delta$  for each  $\sigma$  and  $\delta$  such that

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

- A bijection  $(::_{\delta, \sigma})$  from  $\mathcal{A}^\delta \times \mathcal{A}^\sigma$  to  $\mathcal{A}^{\delta \times \sigma}$  for each  $\delta$  and  $\sigma$ , the inverse of which consists of the projection functions  $\langle \text{Car}_{\delta, \sigma}, \text{Cdr}_{\delta, \sigma} \rangle$ .

We often omit type annotations for the operators  $(\star)$ ,  $(::)$ ,  $\text{Car}$ , and  $\text{Cdr}$ . For any stream pre-model, we define the  $i$ -th projection function  $\pi_i : \mathcal{A}^\sigma \rightarrow \mathcal{A}^{(\sigma)_i}$  for each  $i \geq 0$  as  $\pi_i(s) = \text{Car}(\text{Cdr}^i s)$ .

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  $x^\delta \in \text{TV}$  and  $\rho(\alpha^\sigma) \in \mathcal{A}^\sigma$  for each  $\alpha^\sigma \in \text{SV}$ . The set of environments on  $\mathcal{A}$  is denoted by  $\text{Env}^\mathcal{A}$ .

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

- $\llbracket x^\delta \rrbracket_\rho = \rho(x^\delta)$  and  $\llbracket \alpha^\sigma \rrbracket_\rho = \rho(\alpha^\sigma)$ .

- $\llbracket \lambda x^{\delta'}. t^{\sigma \rightarrow \delta} \rrbracket_\rho \in \mathcal{A}^{\delta' \times \sigma \rightarrow \delta}$ , and  $\llbracket \lambda x^{\delta'}. t^{\sigma \rightarrow \delta} \rrbracket_\rho \star s = \llbracket t \rrbracket_{\rho[x \mapsto \text{Car } s]} \star (\text{Cdr } s)$  for any  $s \in \mathcal{A}^{\delta' \times \sigma}$ ,
- $\llbracket t^{\delta' \times \sigma \rightarrow \delta} u^{\delta'} \rrbracket_\rho \in \mathcal{A}^{\sigma \rightarrow \delta}$ , and  $\llbracket t^{\delta' \times \sigma \rightarrow \delta} u^{\delta'} \rrbracket_\rho \star s = \llbracket t \rrbracket_\rho \star (\llbracket u \rrbracket_\rho :: s)$  for any  $s \in \mathcal{A}^\sigma$ ,
- $\llbracket \mu \alpha^\sigma. t^\delta \rrbracket_\rho \in \mathcal{A}^{\sigma \rightarrow \delta}$ , and  $\llbracket \mu \alpha^\sigma. t^\delta \rrbracket_\rho \star s = \llbracket t \rrbracket_{\rho[\alpha \mapsto s]}$  for any  $s \in \mathcal{A}^\sigma$ ,
- $\llbracket \text{car } S^{\delta \times \sigma} \rrbracket_\rho = \text{Car } \llbracket S \rrbracket_\rho$ ,
- $\llbracket t^{\sigma \rightarrow \delta} S^\sigma \rrbracket_\rho = \llbracket t \rrbracket_\rho \star \llbracket S \rrbracket_\rho$ ,
- $\llbracket t^\delta :: S^\sigma \rrbracket_\rho = \llbracket t \rrbracket_\rho :: \llbracket S \rrbracket_\rho$ ,
- $\llbracket \text{cdr } S^{\delta \times \sigma} \rrbracket_\rho = \text{Cdr } \llbracket S \rrbracket_\rho$ .

4. A stream pre-model  $\mathcal{A}$  is called *full* if  $\mathcal{A}^{\sigma \rightarrow \delta}$  is the whole function space from  $\mathcal{A}^\sigma$  to  $\mathcal{A}^\delta$  for any  $\sigma$  and  $\delta$ , and  $\mathcal{A}^\sigma$  is  $\prod_{i \in \mathbb{N}} \mathcal{A}^{(\sigma)_i}$  for any  $\sigma$ .

**Example 3.2** 1. Let  $\mathcal{A}^X$  be an arbitrary non-empty set for each base type  $X$ , and then we can define a full stream model by  $\mathcal{A}^{\sigma \rightarrow \delta} = \mathcal{A}^\sigma \rightarrow \mathcal{A}^\delta$ , which is the whole function space, and  $\mathcal{A}^\sigma = \prod_{i \in \mathbb{N}} \mathcal{A}^{(\sigma)_i}$ .

2. An example of non-full stream model is the pointed CPO model, in which  $\mathcal{A}^{\sigma \rightarrow \delta}$  is the set of the continuous functions and  $\mathcal{A}^\sigma$  is similarly defined as  $\prod_{i \in \mathbb{N}} \mathcal{A}^{(\sigma)_i}$ .

In these examples,  $\mathcal{A}^\sigma$  is canonically defined as  $\prod_{i \in \mathbb{N}} \mathcal{A}^{(\sigma)_i}$ , but  $\mathcal{A}^\sigma$  is not necessarily isomorphic to  $\prod_{i \in \mathbb{N}} \mathcal{A}^{(\sigma)_i}$ . One example is the following open term model, in which, from an infinite tuple  $\langle t_0, t_1, \dots \rangle$ , we cannot construct a stream  $S$  such that  $\text{cadr}^i(S) = \Lambda_{\mu_{\text{cons}}} t_i$  for any  $i \geq 0$ . Moreover, the mapping  $\langle \pi_i \rangle_{i \in \mathbb{N}}$  from  $\mathcal{A}^\sigma$  to  $\prod_{i \in \mathbb{N}} \mathcal{A}^{(\sigma)_i}$  is not necessarily injective. However, we can assume injectivity without any change of the following discussion, since the mapping  $[S] \mapsto \langle [\text{cadr}^i(S)] \rangle_{i \in \mathbb{N}}$  is injective even in the open term model (Lemma 4.1).

**Proposition 3.3** (Open term model) Define the sets

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

and the functions

$$\begin{aligned} [t] \star [S] &= [tS] & [t] :: [S] &= [t :: S] \\ \text{Car}[S] &= [\text{car } S] & \text{Cdr}[S] &= [\text{cdr } S]. \end{aligned}$$

This structure  $\mathcal{T}$  is a stream model with the meaning function given by  $\llbracket t \rrbracket_\rho = [t\theta_\rho]$ , 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  $(\eta_S)$ . The conditions for the meaning function are proved similarly to the untyped case.  $\square$

**Theorem 3.4** (Soundness and completeness) For any  $t$  and  $u$ ,  $\Gamma; \Delta \vdash t = u : \delta$  holds for some  $\Gamma, \Delta$ , and  $\delta$  if and only if  $\llbracket t \rrbracket_\rho = \llbracket u \rrbracket_\rho$  holds for any stream model  $\mathcal{A}$  and  $\rho \in \text{Env}^\mathcal{A}$ .

**Proof.** ( $\Rightarrow$ ) The soundness is proved by induction on  $\Gamma; \Delta \vdash t = u : \delta$ . We will give only the case of  $(\beta_T)$ .

Case  $(\beta_T)$   $(\lambda x^{\delta'} . t^{\sigma \rightarrow \delta}) u^{\delta'}$ . We have for any  $s \in \mathcal{A}^\sigma$

$$\begin{aligned} \llbracket (\lambda x.t)u \rrbracket_\rho \star s &= \llbracket \lambda x.t \rrbracket_\rho \star (\llbracket u \rrbracket_\rho :: s) \\ &= \llbracket t \rrbracket_{\rho[x \mapsto \llbracket u \rrbracket_\rho]} \star s \\ &= \llbracket t[x := u] \rrbracket_\rho \star s, \end{aligned}$$

where the last line follows from the substitution lemmas:

$$\llbracket t[x := u] \rrbracket_\rho = \llbracket t \rrbracket_{\rho[x \mapsto \llbracket u \rrbracket_\rho]} \quad \llbracket t[\alpha := S] \rrbracket_\rho = \llbracket t \rrbracket_{\rho[\alpha \mapsto \llbracket S \rrbracket_\rho]},$$

which are proved by induction on  $t$  in a straightforward way. Then, by the extensionality, we have  $\llbracket (\lambda x.t)u \rrbracket_\rho = \llbracket t[x := u] \rrbracket_\rho$ .

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

One conclusion of the soundness (and the existence of a non-trivial stream model) is that  $(\text{cdr}^n \alpha)^\sigma =_{\Lambda\mu_{\text{cons}}} (\text{cdr}^m \beta)^\sigma$  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 and an environment which distinguishes the two streams  $\text{cdr}^n \alpha$  and  $\text{cdr}^m \beta$ .

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

**Theorem 3.5** (Consistency) There exist closed  $\Lambda\mu_{\text{cons}}$ -terms  $t$  and  $u$  such that  $\vdash t : \delta$  and  $\vdash u : \delta$  hold, and  $\vdash t =_{\Lambda\mu_{\text{cons}}} u$  does not hold.

**Proof.** Since we have a non-trivial stream model containing two distinct elements, we have  $\lambda xy.x \neq \lambda xy.y$  in a similar way to the untyped case.  $\square$

## 4 Friedman's Theorem for Typed $\Lambda\mu_{\text{cons}}$

In this section, we will define the logical relations on the stream models for the typed  $\Lambda\mu_{\text{cons}}$ , and by means of them, show that we can extend Friedman's theorem to the typed  $\Lambda\mu_{\text{cons}}$ , that is, the equality in  $\Lambda\mu_{\text{cons}}$  is characterized by any individual full stream model.

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

**Lemma 4.1** For any streams  $S_1$  and  $S_2$ , if  $\text{cadr}^i(S_1) =_{\Lambda\mu_{\text{cons}}} \text{cadr}^i(S_2)$  holds for all  $i \geq 0$ , then we have  $S_1 =_{\Lambda\mu_{\text{cons}}} S_2$ .

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

The logical relations on the stream models are inductively defined on the expanded types as follows.

**Definition 4.2** 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  $X$ ,
- $f_1 \mathcal{R}^{\sigma \rightarrow \delta} f_2$  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  $\text{Car } s_1 \mathcal{R}^\delta \text{Car } s_2$  and  $\text{Cdr } s_1 \mathcal{R}^\sigma \text{Cdr } s_2$ . 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 4.3** Let  $\mathcal{R}$  be a logical relation on stream models  $\mathcal{A}_1$  and  $\mathcal{A}_2$ . For contexts  $\Gamma$  and  $\Delta$ , we write  $\rho_1 \mathcal{R}^{\Gamma; \Delta} \rho_2$  to denote the following:

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

**Proposition 4.4** Let  $\mathcal{R}$  be a logical relation on stream models  $\mathcal{A}_1$  and  $\mathcal{A}_2$ . If  $\Gamma; \Delta \vdash_{\Lambda_{\text{cons}}} t : \delta$  and  $\rho_1 \mathcal{R}^{\Gamma; \Delta} \rho_2$ , then we have  $\llbracket t \rrbracket_{\rho_1}^{\mathcal{A}_1} \mathcal{R}^\delta \llbracket t \rrbracket_{\rho_2}^{\mathcal{A}_2}$ .

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

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

Case (Mu)  $\mu \alpha^{\sigma} . t^{\delta} : \sigma \rightarrow \delta$ . 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  $\llbracket t \rrbracket_{\rho_1[\alpha \mapsto s_1]} \mathcal{R}^\delta \llbracket t \rrbracket_{\rho_2[\alpha \mapsto s_2]}$ , so we have  $(\llbracket \mu \alpha. t \rrbracket_{\rho_1} \star s_1) \mathcal{R}^\delta (\llbracket \mu \alpha. t \rrbracket_{\rho_2} \star s_2)$  for any  $s_1 \mathcal{R}^\sigma s_2$  by the condition of  $\llbracket \cdot \rrbracket$ . Therefore, we have  $\llbracket \mu \alpha. t \rrbracket_{\rho_1} \mathcal{R}^{\sigma \rightarrow \delta} \llbracket \mu \alpha. t \rrbracket_{\rho_2}$ .

Case (SApp)  $t^{\sigma \rightarrow \delta} S^\sigma$ . By the induction hypothesis, we have  $\llbracket t \rrbracket_{\rho_1} \mathcal{R}^{\sigma \rightarrow \delta} \llbracket t \rrbracket_{\rho_2}$  and  $\llbracket S \rrbracket_{\rho_1} \mathcal{R}^\sigma \llbracket S \rrbracket_{\rho_2}$ , so we have  $(\llbracket t \rrbracket_{\rho_1} \star \llbracket S \rrbracket_{\rho_1}) \mathcal{R}^\delta (\llbracket t \rrbracket_{\rho_2} \star \llbracket S \rrbracket_{\rho_2})$ , hence  $\llbracket tS \rrbracket_{\rho_1} \mathcal{R}^\delta \llbracket tS \rrbracket_{\rho_2}$ .

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

Case (Cdr)  $\text{cdr } S^{\delta \times \sigma}$ . By induction hypothesis, we have  $\llbracket S \rrbracket_{\rho_1} \mathcal{R}^{\delta \times \sigma} \llbracket S \rrbracket_{\rho_2}$ , so we have  $\text{Cdr } \llbracket S \rrbracket_{\rho_1} \mathcal{R}^\sigma \text{Cdr } \llbracket S \rrbracket_{\rho_2}$ , hence  $\llbracket \text{cdr } S \rrbracket_{\rho_1} \mathcal{R}^\sigma \llbracket \text{cdr } S \rrbracket_{\rho_2}$ .  $\square$

**Corollary 4.5** (Fundamental lemma) Let  $\mathcal{R}$  be a logical relation on  $\mathcal{A}_1$  and  $\mathcal{A}_2$ . If  $\vdash_{\Lambda\mu_{\text{cons}}} t : \delta$  holds, then we have  $\llbracket t \rrbracket^{\mathcal{A}_1} \mathcal{R}^\delta \llbracket t \rrbracket^{\mathcal{A}_2}$ .

**Definition 4.6** 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  $\delta$  and  $\sigma$ :

$$\begin{aligned} \forall a_1 a_2 a'_2. [a_1 \mathcal{R}^\delta a_2 \text{ and } a_1 \mathcal{R}^\delta a'_2 \Rightarrow a_2 = a'_2], \\ \forall s_1 s_2 s'_2. [s_1 \mathcal{R}^\sigma s_2 \text{ and } s_1 \mathcal{R}^\sigma s'_2 \Rightarrow s_2 = s'_2]. \end{aligned}$$

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  $\delta$  and  $\sigma$ .

**Lemma 4.7** 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  $\llbracket t \rrbracket^{\mathcal{A}_1} = \llbracket u \rrbracket^{\mathcal{A}_1}$  holds, then  $\llbracket t \rrbracket^{\mathcal{A}_2} = \llbracket u \rrbracket^{\mathcal{A}_2}$  holds.

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

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

2. For any  $\rho_2 \in \text{Env}^{\mathcal{A}_2}$  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  $\alpha$ , there exists  $s_1$  such that  $s_1 \mathcal{R} \rho_2(\alpha)$ . Then, we can define  $\rho_1 \in \text{Env}^{\mathcal{A}_1}$  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  $\llbracket t \rrbracket_{\rho_1}^{\mathcal{A}_1} \mathcal{R} \llbracket t \rrbracket_{\rho_2}^{\mathcal{A}_2}$  and  $\llbracket u \rrbracket_{\rho_1}^{\mathcal{A}_1} \mathcal{R} \llbracket u \rrbracket_{\rho_2}^{\mathcal{A}_2}$  by Proposition 4.7. Since  $\llbracket t \rrbracket_{\rho_1}^{\mathcal{A}_1} = \llbracket u \rrbracket_{\rho_1}^{\mathcal{A}_1}$  holds by the assumption, we have  $\llbracket t \rrbracket_{\rho_2}^{\mathcal{A}_2} = \llbracket u \rrbracket_{\rho_2}^{\mathcal{A}_2}$  by functionality.  $\square$

The following theorem gives a characterization of  $=_{\Lambda\mu_{\text{cons}}}$ , corresponding to the well-known Friedman's theorem [6]: the extensional equality in  $\lambda^\rightarrow$  is characterized by an arbitrary individual full type hierarchy with infinite bases.

**Theorem 4.8** (Friedman's theorem for  $\Lambda\mu_{\text{cons}}$ ) Suppose that a stream model  $\mathcal{F}$  is full and all of  $\mathcal{F}^X$  are infinite. Then, for any closed typable  $t$  and  $u$ ,  $t =_{\Lambda\mu_{\text{cons}}} u$  holds if and only if  $\llbracket t \rrbracket^{\mathcal{F}} = \llbracket u \rrbracket^{\mathcal{F}}$  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  $u$ ,  $t =_{\Lambda\mu_{\text{cons}}} u$  holds if  $\llbracket t \rrbracket_\rho^{\mathcal{F}} = \llbracket u \rrbracket_\rho^{\mathcal{F}}$  holds for any  $\rho \in \text{Env}^{\mathcal{F}}$ . Consider the open term model  $\mathcal{T}$ , and then we have that  $t =_{\Lambda\mu_{\text{cons}}} u$  holds if and only if  $\llbracket t \rrbracket_\rho^{\mathcal{T}} = \llbracket u \rrbracket_\rho^{\mathcal{T}}$  holds for any  $\rho$ . 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  $\sigma \rightarrow \delta$ . (Functionality) Suppose  $f \mathcal{R}^{\sigma \rightarrow \delta} f'$  and  $f \mathcal{R}^{\sigma \rightarrow \delta} f''$  for  $f \in \mathcal{F}^{\sigma \rightarrow \delta}$  and  $f', f'' \in \mathcal{T}^{\sigma \rightarrow \delta}$ . 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  $f' \in \mathcal{T}^{\sigma \rightarrow \delta}$ . 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  $f \in \mathcal{F}^\sigma \rightarrow \mathcal{F}^\delta = \mathcal{F}^{\sigma \rightarrow \delta}$  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  $f\mathcal{R}^{\sigma \rightarrow \delta} f'$ , 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  $s\mathcal{R}^{\Pi_{i \in \mathbb{N}} \delta_i} s'_1$  and  $s\mathcal{R}^{\Pi_{i \in \mathbb{N}} \delta_i} s'_2$ . 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  $\text{cadr}^i(S_1) =_{\Lambda\mu_{\text{cons}}} \text{cadr}^i(S_2)$  for any  $i$ . By Lemma 4.1, we have  $S_1 =_{\Lambda\mu_{\text{cons}}} S_2$ , that is,  $s'_1 = s'_2$  holds.

(Surjectivity) Let  $s' \in \mathcal{F}^{\Pi_{i \in \mathbb{N}} \delta_i}$ . 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  $s = \langle a_0, a_1, \dots \rangle \in \Pi_{i \in \mathbb{N}} \mathcal{A}^{\delta_i} = \mathcal{A}^{\Pi_{i \in \mathbb{N}} \delta_i}$ , we have  $\pi_i s = a_i\mathcal{R}^{\delta_i} \pi_i s'$  for any  $i$ , so we have  $s\mathcal{R}^{\Pi_{i \in \mathbb{N}} \delta_i} s'$ .  $\square$

## 5 Extension with Fixed-Point Operators

The recursive stream types are essential if we consider recursive functions in typed setting. For example, consider the  $\text{nth}$  function. Let  $Y$  be a fixed-point combinator in the untyped  $\lambda$ -calculus. The function  $\text{nth}$  is defined in the untyped  $\Lambda\mu_{\text{cons}}$  as

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

where  $\text{ifzero}$  and  $\text{pred}$  are supposed to be defined on the Church numerals, and then we have  $\text{nth } S \bar{k} =_{\Lambda\mu_{\text{cons}}} \text{cadr}^k(S)$ , where  $\bar{k}$  is the Church numeral representing  $k$ . The function  $\text{nth}$  can be typed with help of the recursive structure of stream types under the assumption of existence of typed fixed-point operators (and the type of the natural numbers), which we will give in the following.

Suppose that we have coercion operators  $\underline{\delta}$  from term types to stream types and  $\underline{t}$  from terms to streams with its left inverse  $S_\circ$  such that

$$\frac{\Gamma \mid \Delta \vdash t : \delta}{\Gamma \mid \Delta \vdash \underline{t} : \underline{\delta}}, \quad \frac{\Gamma \mid \Delta \vdash S : \underline{\delta}}{\Gamma \mid \Delta \vdash S_\circ : \delta}, \quad \text{and} \quad (\underline{t})_\circ = t.$$

We use the notation  $\mu \underline{x}^\delta. t = \mu \alpha^\delta. t[x^\delta := (\alpha^\delta)_\circ]$ , and then we have  $(\mu \underline{x}. t) \underline{u} = t[x := u]$ . Note that this assumption holds if we have the unit type 1 with a constant  $\vec{()}$  of the type  $[1]$  by defining

$$\underline{\delta} = \delta \times [1] \quad \underline{t} = t :: \vec{()} \quad S_\circ = \text{car } S.$$

**Definition 5.1** The calculus  $\Lambda\mu_{\text{cons}}^{\text{fix}}$  is an extension of the typed  $\Lambda\mu_{\text{cons}}$  with operations  $\underline{\delta}$ ,  $\underline{t}$ ,  $S_\circ$  satisfying the above conditions, and the constants  $\text{fix}_\delta$  of type  $(\underline{\delta} \rightarrow \delta) \rightarrow \delta$  for each  $\delta$  such that

$$\text{fix}_\delta \underline{f} = \underline{f(\text{fix}_\delta \underline{f})} \quad \text{for any } f : \underline{\delta} \rightarrow \delta.$$

Note that an untyped term satisfying this condition for  $\text{fix}_\delta$  is  $\mu \underline{f}. (\lambda x. f(\underline{xx})) (\lambda x. f(\underline{xx}))$ .

If we have the type  $N$  of the natural number with basic operators on  $N$  such as  $\text{pred} : N \rightarrow N$  and  $\text{ifzero} : N \times \delta \times \underline{\delta} \rightarrow \delta$ , we can define  $\text{nth}$  in  $\Lambda\mu_{\text{cons}}^{\text{fix}}$  as  $\text{fix}_\delta(F)$ , where  $\delta = [\delta_0] \rightarrow N \rightarrow N$  and

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

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

The calculus  $\Lambda\mu_{\text{cons}}^{\text{fix}}$  is sound with respect to the following simple modification of the stream models.

**Definition 5.2** (Stream Models for  $\Lambda\mu_{\text{cons}}^{\text{fix}}$ ) We call  $\mathcal{A}$  is a stream model for  $\Lambda\mu_{\text{cons}}^{\text{fix}}$  if  $\mathcal{A}$  is a stream model for the typed  $\Lambda\mu_{\text{cons}}$  with the following additional structures.

- An embedding  $(\cdot) : \mathcal{A}^\delta \rightarrow \mathcal{A}^{\underline{\delta}}$  and its left inverse  $(\cdot)_\circ : \mathcal{A}^{\underline{\delta}} \rightarrow \mathcal{A}^\delta$  such that  $(\underline{d})_\circ = d$  for any  $d \in \mathcal{A}^\delta$ .
- $\text{Fix}_\delta := \llbracket \text{fix}_\delta \rrbracket^{\mathcal{A}}$  is an element of  $\mathcal{A}^{(\underline{\delta} \rightarrow \delta) \rightarrow \delta}$  such that, for any  $f \in \mathcal{A}^{\underline{\delta} \rightarrow \delta}$ ,

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

**Example 5.3** An example of the stream model for  $\Lambda\mu_{\text{cons}}^{\text{fix}}$  can be constructed as the pointed CPO model.

- $\mathcal{D}^1$  and  $\mathcal{D}^N$  are the flat domains constructed from  $\{\bullet\}$  (with a fixed element  $\bullet$ ) and  $\mathbb{N}$ , respectively.
- $\mathcal{D}^{\sigma \rightarrow \delta} := [\mathcal{D}^\sigma \rightarrow \mathcal{D}^\delta]$  is the set of continuous functions, and  $\mathcal{D}^\sigma := \prod_{i \in \mathbb{N}} \mathcal{D}^{(\sigma)_i}$ .
- $\mathcal{D}^{\underline{\delta}} = \mathcal{D}^\delta \times \mathcal{D}^{[1]}$ ,  $\underline{d} = \langle d, \bullet, \bullet, \dots \rangle$ , and  $s_\circ = \text{Car } s$ .
- For any  $p \in \mathcal{D}^{\underline{\delta} \rightarrow \delta} = \mathcal{D}^{\underline{\delta} \rightarrow \delta} \times \mathcal{D}^{[1]}$ ,

$$\text{Fix}_\delta(p) := \bigsqcup_{n \in \mathbb{N}} (\bar{\lambda} d \in \mathcal{D}^\delta. (\text{Car } p)(\underline{d})^n(\perp)).$$

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

$$\text{Fix}_\delta(\underline{f}) = \bigsqcup_{n \in \mathbb{N}} (\bar{\lambda} d \in \mathcal{D}^\delta. f(\underline{d}))^n(\perp),$$

and then we have  $f(\text{Fix}_\delta(\underline{f})) = \text{Fix}_\delta(\underline{f})$ .

## 6 Related 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 [11, 15], Parigot's original  $\lambda\mu$ -calculus in [12], and Gaboardi and Saurin's  $\Lambda_{\mathcal{S}}$  in [7].

### 6.1 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,  $\Lambda\mu_{\text{cons}}$  is not less than  $\Lambda_{\mathcal{S}}$ . Indeed, we can give a translation from  $\Lambda_{\mathcal{S}}$  to  $\Lambda\mu_{\text{cons}}$  preserving typability.

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

$$\mathcal{T} ::= o \mid \mathcal{T} \rightarrow \mathcal{T} \mid \mathcal{S} \Rightarrow \mathcal{T} \qquad \mathcal{S} ::= \perp \mid \mathcal{T} \rightarrow \mathcal{S},$$

and  $\mathcal{T}$  and  $\mathcal{S}$  correspond to term types and stream types in  $\Lambda\mu_{\text{cons}}$ , respectively. We consider types modulo the equivalence defined from the axiom  $(\mathcal{T} \rightarrow \mathcal{S}) \Rightarrow \mathcal{T}' \equiv \mathcal{T} \rightarrow (\mathcal{S} \Rightarrow \mathcal{T}')$ . The stream type of the form  $\mathcal{T} \rightarrow \mathcal{S}$  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  $\Lambda_{\mathcal{S}}$  also seems natural under correspondence between streams and (evaluation) contexts. For example, a stream constructed from a head  $t$  and a tail  $\alpha$  is represented as a context  $\llbracket t \alpha \rrbracket$  in  $\Lambda_{\mathcal{S}}$ , 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  $\alpha$  has a type  $\mathcal{S}$ , the context should have the type  $\mathcal{T} \rightarrow \mathcal{S}$  as [3].

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

**Definition 6.1** Fix a type variable  $O$  in  $\Lambda\mu_{\text{cons}}$ . We define the translation  $\overline{(\cdot)}$  from the  $\Lambda_{\mathcal{S}}$ -types to the  $\Lambda\mu_{\text{cons}}$ -types by

$$\begin{aligned} \overline{o} &= [O] \rightarrow O & \overline{\mathcal{T} \rightarrow \mathcal{T}'} &= \overline{\mathcal{T}} \times \overline{\mathcal{T}'}_s \rightarrow \overline{\mathcal{T}'}_t & \overline{\mathcal{S} \Rightarrow \mathcal{T}} &= \overline{\mathcal{S}} \rightarrow \overline{\mathcal{T}} \\ \overline{\perp} &= [O] & \overline{\mathcal{T} \rightarrow \mathcal{S}} &= \overline{\mathcal{T}} \times \overline{\mathcal{S}}, \end{aligned}$$

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

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

**Example 6.3** Pagani and Saurin showed in [11] 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

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

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

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

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  $((A \rightarrow B) \rightarrow A) \rightarrow A$ , 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. This gap 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 [15] and van Bakel et al. [18] 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 [15] and the translation in Definition 6.1. It corresponds to the continuation-passing-style translation of Thielecke [17].

**Definition 6.4** (Negative translation) We fix a type variable  $O$  and an arbitrary stream type  $\theta$  (for example, we take  $\theta = [O]$ ). We write  $\neg\sigma$  for  $\sigma \rightarrow O$ . The negative translations  $\overline{(\cdot)}$  from the implicational formulas to term types in  $\Lambda\mu_{\text{cons}}$  and  $(\cdot)^\bullet$  from the implicational formulas to stream types in  $\Lambda\mu_{\text{cons}}$  are defined as

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

**Proposition 6.5** If  $\Gamma \vdash t : A; \Delta$  holds in the Parigot's original typed  $\lambda\mu$ -calculus in [12], then  $\neg\Gamma^\bullet; \Delta^\bullet \vdash t : \neg A^\bullet$  holds in  $\Lambda\mu_{\text{cons}}$ .

As a corollary of the proposition, every typable  $\lambda$ -term in the simply-typed  $\lambda$ -calculus, denoted by  $\lambda^\rightarrow$ , is typable in  $\Lambda\mu_{\text{cons}}$ . Moreover, for the  $\lambda$ -terms, we have a more direct translation on types, which is similar to the translation given in [18]. Fix an arbitrary type variable  $O$ , and define the translation on the simple types as

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

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

Hence, regarding typability,  $\Lambda\mu_{\text{cons}}$  can be considered to include  $\lambda^\rightarrow$ ,  $\lambda\mu$ , and  $\Lambda\mu_{\mathcal{S}}$ . On the other hand,  $\Lambda\mu_{\text{cons}}$  can type properly wider class of terms than the other systems due to the recursive types, that is,  $\Lambda\mu_{\text{cons}}$  is not conservative over  $\lambda^\rightarrow$  with respect to typability. An example is the following. Let  $\delta = [X] \rightarrow X$  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  $\delta$  under the context  $\Gamma, w : \delta \times \delta \times [X] \rightarrow X, x : \delta \times [X] \rightarrow X$  in  $\Lambda\mu_{\text{cons}}$ . 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  $\lambda$ -term, and hence it shows the typability in  $\Lambda\mu_{\text{cons}}$  is not conservative over  $\lambda^\rightarrow$ .

**Proposition 6.6** Every typable term in either  $\lambda^\rightarrow$ ,  $\lambda\mu$ , or  $\Lambda_{\mathcal{S}}$  is typable in  $\Lambda\mu_{\text{cons}}$ . On the other hand, there is a  $\lambda$ -term which is typable in  $\Lambda\mu_{\text{cons}}$ , and not typable in neither  $\lambda^\rightarrow$ ,  $\lambda\mu$ , nor  $\Lambda_{\mathcal{S}}$ .

## 6.2 Gaboardi and Saurin's $\Lambda_{\mathcal{S}}$

Gaboardi and Saurin's  $\Lambda_{\mathcal{S}}$  [7] considers more general recursive types. We can define a translation from  $\Lambda\mu_{\text{cons}}$  to  $\Lambda_{\mathcal{S}}$  preserving typability as follows:

$$\begin{array}{ll}
\overline{\overline{X}} = X & \overline{\overline{\sigma \rightarrow \delta}} = \overline{\overline{\sigma}} \rightarrow \overline{\overline{\delta}} \\
\overline{\overline{[\delta_0, \dots, \delta_{n-1}]}} = \mu \mathcal{X}. \delta_0 \rightarrow \dots \rightarrow \delta_{n-1} \rightarrow \mathcal{X} & \overline{\overline{\delta \times \sigma}} = \overline{\overline{\delta}} \rightarrow \overline{\overline{\sigma}} \\
\overline{\overline{x}} = x & \overline{\overline{\alpha}} = \alpha \\
\overline{\overline{\lambda x. t}} = \lambda x. \overline{\overline{t}} & \overline{\overline{t :: S}} = [\overline{\overline{t}}; \overline{\overline{S}}] \\
\overline{\overline{tu}} = \overline{\overline{t}} \overline{\overline{u}} & \overline{\overline{\text{cdr } S}} = \sigma((\lambda x. \mu \alpha. \tau(\alpha)) \overline{\overline{S}}) \\
\overline{\overline{\mu \alpha. t}} = \mu \alpha. \overline{\overline{t}} & \\
\overline{\overline{tS}} = \overline{\overline{t}} \overline{\overline{S}} & \\
\overline{\overline{\text{car } S}} = (\lambda x. \mu \alpha. x) \overline{\overline{S}} &
\end{array}$$

**Proposition 6.7** 1. If  $\Gamma; \Delta \vdash_{\Lambda\mu_{\text{cons}}} t : \delta$ , then  $\overline{\overline{\Gamma}} \vdash_{\Lambda_{\mathcal{S}}} \overline{\overline{t}} : \overline{\overline{\delta}}; \overline{\overline{\Delta}}$ .

2. If  $t = u$  is derivable in  $\Lambda\mu_{\text{cons}}$ , then  $\overline{\overline{t}} = \overline{\overline{u}}$  in  $\Lambda_{\mathcal{S}}$ .

Gaboardi and Saurin also 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 left inverse, which are denoted by `tm` and `st` in this paper, with typing rules

$$\frac{S : \sigma}{\text{tm}(S) : \text{Str}(\sigma)} \quad \frac{t : \text{Str}(\sigma)}{\text{st}(t) : \sigma},$$

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

With the combination of the coercion operators of  $\Lambda_{\mathcal{S}}$  and the fixed-point operator discussed in the previous section, we can define recursive functions returning streams. An example is the function `alt` interleaving two streams as

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

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

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

and then `alt` has the type  $\delta$ .

## 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] 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.
- [4] R. David and W. Py.  $\lambda\mu$ -calculus and Böhm’s theorem. *The Journal of Symbolic Logic*, 66:407–413, 2001.
- [5] U. de’Liguoro. The approximation theorem for the  $\Lambda\mu$ -calculus. *Mathematical Structures in Computer Science*, to appear.
- [6] H. Friedman. Equality between functionals. In Parikh, R., editor, *Logic Colloquium*, volume 453 of *Lecture Notes in Mathematics*, pages 22–37. Springer, 1973.
- [7] M. Gaboardi and A. Saurin. A foundational calculus for computing with streams. In *12th Italian Conference on Theoretical Computer Science*, 2010.
- [8] R. Milner. A complete inference system for a class of regular behaviours. *Journal of Computer and System Sciences*, 28:439–466, 1984.
- [9] 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.
- [10] K. Nakazawa and T. Nagai. Reduction system for extensional lambda-mu calculus. In *25th International Conference on Rewriting Techniques and Applications joint with the 12th International Conference on Typed Lambda Calculi and Applications (RTA-TLCA 2014)*, to appear.
- [11] M. Pagani and A. Saurin. Stream associative nets and  $\lambda\mu$ -calculus. Research Report 6431, INRIA, 2008.
- [12] 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.
- [13] A. Salomaa. Two complete axiom systems for the algebra of regular events. *Journal of the Association for Computing Machinery*, 13(1):158–169, 1966.
- [14] 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.
- [15] A. Saurin. Typing streams in the  $\Lambda\mu$ -calculus. *ACM Transactions on Computational Logic*, 11, 2010.
- [16] T. Streicher and B. Reus. Classical logic, continuation semantics and abstract machines. *Journal of Functional Programming*, 8(6):543–572, November 1998.
- [17] H. Thielecke. *Categorical structure of continuation passing style*. PhD thesis, University of Edinburgh, 1997.

- [18] 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.
- [19] S. van Bakel, F. Barbanera, and U. de'Liguoro. Characterisation of strongly normalising  $\lambda\mu$ -terms. In S. Graham-Lengrand and L. Paolini, editors, *Sixth Workshop on Intersection Types and Related Systems (ITRS 2012)*, volume 121, pages 1–17, 2012.