A Hierarchy for Delimited Continuations in Call-by-Name

Alexis Saurin

PPS & INRIA πr^2[⋆]
saurin@pps.jussieu.fr

Abstract. $\Lambda\mu$-calculus was introduced as a Böhm-complete extension of Parigot's $\lambda\mu$-calculus. $\Lambda\mu$-calculus, contrarily to Parigot's calculus, is a calculus of CBN *delimited control* as evidenced by Herbelin and Ghilezan. In their seminal paper on (CBV) delimited control, Danvy and Filinski introduced the CPS Hierarchy of control operators $(\texttt{shift}_i/\texttt{reset}_i)_{i\in\omega}$.

In a similar way, we introduce in the present paper the *Stream Hierarchy*, a hierarchy of calculi extending and generalizing $\Lambda\mu$-calculus. The $(\Lambda^n)_{n\in\omega}$-calculi have Church-Rosser and Böhm theorems. We then present sound and complete CPS translations for the hierarchy. Next, we investigate the operational content of the hierarchy through its abstract machines, the $(\Lambda^n)_{n\in\omega}$-KAM. Finally, we establish that the Stream hierarchy is indeed a CBN analogue to the CPS hierarchy.

Keywords: $\Lambda\mu$-calculus, delimited control, CPS hierarchy, Böhm theorem, CPS translation, Abstract machine, Streams.

1 Introduction

Curry-Howard in Classical Logic, $\lambda\mu$-calculus and Separation. Curry-Howard correspondence [17] was first designed as a correspondence between intuitionistic natural deduction (NJ) and simply typed λ-calculus. The extension of the correspondence to classical logic resulted in strong connections with control operators in functional languages as first noticed [15] by Griffin who analysed the logical interpretation of Felleisen's $\mathcal{C}$ operator [12]. Shortly after Griffin, Parigot introduced $\lambda\mu$-calculus [27] as an extension of λ-calculus corresponding to minimal classical natural deduction [1,26] in which one can encode usual control operators. $\lambda\mu$-calculus became one of the most widely studied classical λ-calculi, both in the typed and untyped setting, for several reasons: it naturally extends λ-calculus while retaining most of λ-calculus standard properties and intuitionistic natural deduction in a straightforward way. However, a fundamental property of pure λ-calculus, known as separation property (or Böhm theorem [6]), does not hold for $\lambda\mu$-calculus [29,9]. In a previous work, we introduced $\Lambda\mu$-calculus, an extension to $\lambda\mu$-calculus, for which we proved that separation holds [31].

[⋆] This work was partially funded by the french foreign office with the support of a *Bourse Lavoisier*.

Delimited control and the CPS hierarchy. Delimited control refers to a class of control operators which are much more expressive than non-delimited control operators (like `call/cc` for instance) in that they allow to simulate various side-effects [13], the monadic side-effects. In their seminal paper on `shift`/`reset` [7], Danvy and Filinski defined `shift`/`reset` delimited-control operators by their CPS semantics. They also introduced a hierarchy of such control operators, $(\mathtt{shift}_i/\mathtt{reset}_i)_{i\in\omega}$, which are obtained by iterating CPS translations and that is known as the CPS hierarchy. Delimited control and the CPS hierarchy found applications in linguistics, normalization by evaluation, partial evaluation or concurrency. While the emphasis was traditionally given to the delimited-control languages in call-by-value, recent works [16,21] have advocated the reasons for studying CBN delimited control.

In this paper, we develop a CBN analogue to the CPS hierarchy, based on $\Lambda\mu$-calculus. We develop our work on the strong connections between $\Lambda\mu$-calculus and calculi with delimited continuations in call-by-name evidenced by Herbelin and Ghilezan [16].

Structure of the Paper. In Section 2, we first review Parigot's $\lambda\mu$-calculus and $\Lambda\mu$-calculus as well as the main properties of those calculi. In Section 3, we motivate and define the $(\Lambda^n)_{n\in\omega}$-calculi which we refer to as the *stream hierarchy*. We establish two essential results of its meta-theory: Church-Rosser and Böhm theorems. Section 4 is concerned with translations of the stream hierarchy into λ-calculus which are sound and complete and we develop in Section 5 Krivine's style abstract machines [23] for the hierarchy. Finally, Section 6 makes precise the relationships between the Stream hierarchy and the CPS hierarchy. A long version of this paper can be found on the author's webpage [30].

2 Background and Notations: From $\lambda\mu$ to $\Lambda\mu$.

In this section, we recall some background on $\Lambda\mu$-calculus: starting with Parigot's $\lambda\mu$, we introduce $\Lambda\mu$-calculus via the property of Separation.

Parigot's Original Calculus: $\lambda\mu$. In 1992, Parigot proposed an extension of λ-calculus providing "an algorithmic interpretation of classical natural deduction" [27]: $\lambda\mu$-calculus is in Curry-Howard correspondence [17] with classical natural deduction [26,27]. Although initially motivated by the correspondence with classical logic, $\lambda\mu$-calculus is now widely studied in its untyped version as we do in the rest of this paper.

Definition 1. **$\lambda\mu$-terms** *$(t, u, v, \dots \in \Sigma_{\lambda\mu})$ are defined by the following syntax:*

$$\boxed{\Sigma_{\lambda\mu} \qquad t, u ::= \; x \mid \lambda x.t \mid (t)u \mid \mu\alpha.(t)\beta}$$

with $x \in \mathcal{V}$ and $\alpha, \beta \in \mathcal{V}_c$, $\mathcal{V}$ and $\mathcal{V}_c$ being two disjoint infinite sets of variables.

In $\mu\alpha.(t)\beta$, variable β is in the scope of $\mu\alpha$. For $t \in \Sigma_{\lambda\mu}$, $(t)\alpha$ is not *in $\Sigma_{\lambda\mu}$, but we refer to such it as a* **named term** *and generically write n (and thus we write $\mu\alpha.n$). The set of closed $\lambda\mu$-terms is denoted by $\Sigma^c_{\lambda\mu}$.*

Remark 1. *The reader may have noticed that we use an alternative notation for $\lambda\mu$-terms that we introduced and justified in previous works [31,33], writing $(t)\alpha$ instead of the more common $[\alpha]t$ (this shall later be extended to the $(\Lambda^i)_{i\in\omega}$).*

In this paper, we shall use Krivine's notation [22] for terms of λ-calculus and its various extensions considered here: we write $(t)u$ for λ-application (instead of (MN)). As usual we consider λ-application to be left-associative, that is $(t)u_1 \dots u_{k-1}u_k$ shall be read as $(\dots((t)u_1)\dots u_{k-1})u_k$. This notation is extended to variables of $\mathcal{V}_c$ (and later on to the variables of the hierarchy). For instance, we shall write $\mu\alpha.(t)u\beta$ instead of $\mu\alpha.((t)u)\beta$.

Definition 2. **$\lambda\mu$-reduction**, *written $\longrightarrow_{\lambda\mu}$, is induced by the following rules:*

$$\boxed{\begin{array}{llll} (\lambda x.t)u \longrightarrow_\beta t\{u/x\} & (\mu\alpha.n)\beta \longrightarrow_\rho n\{\beta/\alpha\} & \\ (\mu\alpha.n)u \longrightarrow_\mu \mu\alpha.n\{(v)u\alpha/(v)\alpha\} & \mu\alpha.(t)\alpha \longrightarrow_\theta t & \textit{if } \alpha \notin FV(t) \end{array}}$$

$n\{(v)u\alpha/(v)\alpha\}$ substitutes (without variable-capture) every named term $(v)\alpha$ in n by $(v)u\alpha$. This substitution is called ***structural substitution*** [27].

A $\lambda\mu$-calculus Satisfying Böhm Theorem: $\Lambda\mu$-calculus. $\lambda\mu$ satisfies standard properties of λ-calculus such as confluence [27,29], subject reduction [27] and SN [28]. However, Böhm theorem fails in $\lambda\mu$-calculus (more precisely in its extensional version, $\lambda\mu\eta$-calculus [29,9]). This led us [31] to define an extension to $\lambda\mu\eta$, $\Lambda\mu$-calculus, for which we proved Böhm theorem: the more liberal syntax of $\Lambda\mu$ makes new contexts available and thus achieves a Böhm Out.

Definition 3. *$\Lambda\mu$-terms ($t, u, v \dots \in \Sigma_{\Lambda\mu}$) are defined by the following syntax:*

$$\boxed{\Sigma_{\Lambda\mu} \qquad t, u ::= \ x \mid \lambda x.t \mid (t)u \mid \mu\alpha.t \mid (t)\alpha}$$

where x (resp. α) ranges over an infinite set $\mathcal{V}_t$ (resp. $\mathcal{V}_s$) of term (resp. stream) variables. $\mathcal{V}_t$ and $\mathcal{V}_s$ are disjoint. The set of closed $\Lambda\mu$-terms is denoted by $\Sigma^c_{\Lambda\mu}$.

Remark 2. *Since $\alpha \notin \Sigma_{\Lambda\mu}$, it is clear that notations $(t)\alpha$ and $(t)u$ are not ambiguous. Notice that $\Sigma_{\lambda\mu} \subsetneq \Sigma_{\Lambda\mu}$ and that named terms of definition 1 are now elements of $\Sigma_{\Lambda\mu}$. Moreover, terms such as $\mu\alpha.\mu\beta.t$ or $\lambda x.(t)\alpha y$ are in $\Sigma_{\Lambda\mu} \setminus \Sigma_{\lambda\mu}$.*

Definition 4. **$\Lambda\mu$-reduction**, *written $\longrightarrow_{\Lambda\mu}$, is induced by the following rules:*

$$\boxed{\begin{array}{llll} (\lambda x.t)u \longrightarrow_{\beta_T} t\{u/x\} & \lambda x.(t)x & \longrightarrow_{\eta_T} t & \textit{if } x \notin FV(t) \\ (\mu\alpha.t)\beta \longrightarrow_{\beta_S} t\{\beta/\alpha\} & \mu\alpha.(t)\alpha & \longrightarrow_{\eta_S} t & \textit{if } \alpha \notin FV(t) \\ & \mu\alpha.t & \longrightarrow_{fst} \lambda x.\mu\alpha.t\{(v)x\alpha/(v)\alpha\} & \textit{if } x \notin FV(t) \end{array}}$$

Remark 3. *Notice that μ is not part of $\Lambda\mu$-calculus reduction system. It can indeed be simulated by a sequence of fst and β_T-reduction; see [31,33] for details. Names for reductions in $\Lambda\mu$ come from the stream interpretation of $\Lambda\mu$: $\mathcal{V}_S$-variables are place-holders for streams of $\Lambda\mu$-terms; see next section for details.*

The Böhm theorem for $\Lambda\mu$ is stated with respect to a set of *canonical normal forms* (corresponding to $\beta\eta$-normal forms in λ-calculus), which are terms in $\beta_T\eta_T\beta_S\eta_S$-normal form such that no *fst*-reduction step creates a non-*fst* redex:

Definition 5. *A $\Lambda\mu$-term t is in* **canonical normal form (CNF)** *if it is $\beta_T\eta_T\beta_S\eta_S$-normal and if it contains no subterm of the form $(\lambda x.u)\alpha$ nor $(\mu\alpha.u)v$.*

Theorem 4 (Böhm theorem [31]). *Let $t, t' \in \Sigma^c_{\Lambda\mu}$ in CNF. If $t \neq_{\Lambda\mu} t'$, then there exists a context*[1] $C[]$ *st.* $C[t] \longrightarrow^{\star}_{\Lambda\mu} \lambda x.\lambda y.x$ *and* $C[t'] \longrightarrow^{\star}_{\Lambda\mu} \lambda x.\lambda y.y$.

Confluence holds in $\Lambda\mu$ [32,34] under the same hypothesis as in $\lambda\mu\eta$-calculus:

Theorem 5. $\forall t, t', t'' \in \Sigma^c_{\Lambda\mu}, \exists u \in \Sigma_{\Lambda\mu}$ *s.t.* $t \longrightarrow^{\star}_{\Lambda\mu} t', t'' \Rightarrow t', t'' \longrightarrow^{\star}_{\Lambda\mu} u$.

3 λ, μ and Beyond: The Stream Hierarchy

In the present section, we introduce the $(\Lambda^n)_{n\in\omega}$-calculi that we refer to as the *stream hierarchy*. This hierarchy of calculi is intended to be a call-by-name analogous to the CPS hierarchy. We first motivate our approach before defining the hierarchy and focusing on the metatheory of $(\Lambda^n)_{n\in\omega}$-calculi (they satisfy confluence and separation). In the following sections, we shall then study CPS translations and abstract machines for the hierarchy and finally, we shall establish that the Stream Hierarchy is indeed a CBN analogue to the CPS hierarchy in the final section of the paper.

3.1 Motivating the Stream Hierarchy

$\Lambda\mu$-calculus, a CBN calculus of delimited control. Separation theorem for $\Lambda\mu$-calculus can be seen as a consequence of the fact that $\Lambda\mu$-calculus admits more contexts than Parigot's $\lambda\mu$. As a consequence, it allows for a more powerful exploration of terms. Typical contexts used in the separation proofs are $[]u_1 \dots u_m\beta_u v_1 \dots v_n\beta_v$. This exploits the fact that a context of the form $[]u_1 \dots u_m\beta_u$ *delimits* the part of the environment that can be passed through the left-most μ-abstracted variable (*i.e.* α) when term $\mu\alpha.\mu\alpha'.t$ is placed in the hole. As a result, one can access to the second μ-abstracted variable α' thanks to the second portion of the context, $v_1 \dots v_n\beta_v$.

Based on this fact, Herbelin and Ghilezan [16] evidenced strong connections between $\Lambda\mu$-calculus and calculi with delimited continuations in the spirit of Danvy and Filinski `shift`/`reset` operators [7] using the calculus $\lambda\mu\widehat{\mathtt{tp}}$. In its call-by-value version, $\lambda\mu\widehat{\mathtt{tp}}$ is equivalent to Danvy-Filinski's `shift`/`reset` operators while in its call-by-name version the calculus is equationally correspondent to $\Lambda\mu$-calculus. This led Herbelin & Ghilezan to assert that $\Lambda\mu$-calculus is a CBN calculus of delimited control.

[1] The context may be asked to be "stream applicative", *ie.* of the form: $[]t_{1,1} \dots t_{1,n_1}\alpha_1 \dots t_{k,1} \dots t_{k,n_k}\alpha_k$.

CPS Hierarchy. In their seminal paper on `shift`/`reset` [7], Danvy and Filinski introduced a hierarchy of control operators, $(\mathtt{shift}_i/\mathtt{reset}_i)_{i\in\omega}$, which are obtained by iterated CPS translations. This is known as the *CPS hierarchy*. In the following, we shall refer to it as the CPS hierarchy or $\lambda\mathcal{S}_n$ and adopt Kameyama's terminology [19]:

Definition 6 ($\lambda\mathcal{S}_n$)

$$\boxed{\begin{array}{lrll}
\Sigma_{\lambda\mathcal{S}_n} & t,u ::= & x \mid \lambda x.t \mid (t)u \mid \langle t\rangle_i \mid \mathcal{S}_i k.t & 1\leq i\leq n\\
& E_v^i ::= & [] \mid (E_v^i)t \mid (V)E_v^i \mid \langle E_v^i\rangle_j & 1\leq j\leq i\\
& V ::= & x \mid \lambda x.t & \\
\\
& (\lambda x.t)V & \longrightarrow t\{V/x\} & \\
& \langle V\rangle_i & \longrightarrow V & \\
& \langle E_v^{j-1}[\mathcal{S}_j k.t]\rangle_i & \longrightarrow \langle t\left\{\lambda x.\langle E_v^{j-1}[x]\rangle_j/k\right\}\rangle_i &
\end{array}}$$

While the emphasis was traditionally given to the delimited-control languages in call-by-value, recent works have advocated the interest of studying call-by-name delimited control [16,21], although CBN delimited control behaves quite differently from call-by-value. In particular, in pursuing the investigation of call-by-name delimited control, it is quite natural to wonder whether an analogous to the CPS hierarchy exists in the call-by-name world.

$\Lambda\mu$-calculus, Streams and Infinitary λ-calculi. The *fst*-rule allows for an operational interpretation of $\Lambda\mu$-calculus as a stream calculus with the ability to abstract over streams of $\Lambda\mu$-terms. With this interpretation of $\mathcal{V}_S$-variables as place-holders for streams of $\Lambda\mu$-terms:

– the effect of the *fst*-rule is to instantiate the *first* elements of a stream:

$$\mu\alpha.t \longrightarrow^{\star}_{fst} \lambda x_1\ldots\lambda x_n.\mu\alpha.t\{(v)x_1\ldots x_n\alpha/(v)\alpha\}$$

– $\mu\alpha$ is considered as an ***abstraction over streams of terms*** $(\lambda x_1^\alpha\ldots x_n^\alpha\ldots.t)$ while $(t)\alpha$ can be seen as the construction ***passing a stream as an argument*** to t $((t)x_1^\alpha\ldots x_n^\alpha\ldots)$;
– β_S and η_S are respectively the corresponding of β-reduction and η-reduction for streams (or an infinite reduction sequence of β, resp η) and rule *fst* corresponds to popping the first element of a stream (or matching it);
– actually, $\Lambda\mu$-calculus can be seen as a core functional language for stream, this direction being investigated in a current work with M. Gaboardi (see long version of the paper for details).

Parigot already noticed some (weak) form of this in his seminal paper where "the operator μ looks like a λ having potentially infinite number of arguments" [27]. Viewing μ as an operator iterating λ-abstraction until limit ordinal ω, the parallel with infinitary λ-calculi is natural. Such infinitary calculi have been considered in the literature [3,4,20] both to study infinite structures arising in lazy languages or to study consistency problems in λ-calculus. Though, infinitary λ-calculi have been designed in a much different way from the infinitary calculus underlying $\Lambda\mu$-calculus: while a reduction sequence may have transfinite length, depths of terms are bounded by ω (that is any subterm of an infinite term is at finite depth): subterms at transfinite depths are considered meaningless. On the contrary, with $\Lambda\mu$-calculus, limit ordinal ω is reached by one μ-abstraction which is a limit ordinal construction: $\mu\alpha.\mu\beta.\lambda x.x$ would correspond to transfinite term $\lambda x_0, x_1 \dots x_\omega, x_{\omega+1} \dots x_{\omega 2}.x_{\omega 2}$ in which $\lambda x_{\omega 2}.x_{\omega 2}$ is at depth $\omega 2$.

Even though we will not pursue this direction in this paper, this theme has been extremely influential in developing the stream hierarchy. Indeed, once a transfinite calculus is unveiled, the question of the ordinal by which it is indexed (if any) is pending: λ-calculus corresponds to ordinal ω while $\Lambda\mu$-calculus corresponds to ordinal ω^2 but what about other ordinals such as ω^3 for instance? The stream hierarchy is actually related to this question.

3.2 Definition of the Hierarchy of $(\Lambda^n)_{n\in\omega}$-calculi

Definition 7. *Let $\mathcal{V}$ be a countable set of variables ($x, y, \dots \in \mathcal{V}$). For any $i \in \omega$, one considers a copy of $\mathcal{V}$, named $\mathcal{V}^i$ ($x^i, y^i, \dots$ denoting the elements of $\mathcal{V}^i$), those copies being pairwise disjoint. Λ^ω-terms ($t, u, v, \dots \in \Sigma_{\Lambda^\omega}$) are defined by the following grammar (closed Λ^ω-terms are denoted by $\Sigma^c_{\Lambda^\omega}$):*

$$\boxed{\begin{array}{lllll} \Sigma_{\Lambda^\omega} & t,u ::= x^0 & \mid \lambda^0 x.t & \mid (t)u & \\ & & \mid \lambda^i x.t & \mid (t)x^i & \textit{for any } i>0 \end{array}}$$

In $\lambda^i x.t$ (resp x^i), i is the **level** *of the abstraction (resp. variable) and $\lambda^i x$ binds every variable x^i which is free in t. An α-equivalence straightforwardly follows.*

Definition 8. *For $n \in \omega$, Σ_{Λ^n} (resp. $\Sigma^c_{\Lambda^n}$) is the restriction of Σ_{Λ^ω} (resp. $\Sigma^c_{\Lambda^\omega}$) to terms with binders and variables of level lower or equal to n, for $i \le n$.*

Definition 9. *For $n \in \omega$, $\longrightarrow_{\Lambda^n}$ is the reduction on Σ_{Λ^n} induced by rules:*

$$\boxed{\begin{array}{llll} (\lambda^0 x.t)u & \longrightarrow_{\beta^0} & t\left\{u/x^0\right\} & \\ (\lambda^i x.t)y^i & \longrightarrow_{\beta^i} & t\left\{y^i/x^i\right\} & \textit{if } 0<i\le n \\ (\lambda^i x.t)u & \longrightarrow_{\mu^{i/0}} & \lambda^i x.t\left\{(v)ux^i/(v)x^i\right\} & \textit{if } 0<i\le n \\ (\lambda^i x.t)y^j & \longrightarrow_{\mu^{i/j}} & \lambda^i x.t\left\{(v)y^j x^i/(v)x^i\right\} & \textit{if } 0<j<i\le n \end{array}}$$

Definition 10. *For $n \in \omega$, $\longrightarrow_{\Lambda^n_\eta}$ is the reduction on Σ_{Λ^n} induced by rules:*

$$\boxed{\begin{array}{llll} (\lambda^0 x.t)u & \longrightarrow_{\beta^0} & t\left\{u/x^0\right\} & \\ (\lambda^i x.t)y^i & \longrightarrow_{\beta^i} & t\left\{y^i/x^i\right\} & \textit{if } 0<i\le n \\ \lambda^i x.(t)x^i & \longrightarrow_{\eta^i} & t & \textit{if } x^i \notin FV(t), 0\le i\le n \\ \lambda^i x.t & \longrightarrow_{fst^{i/j}} & \lambda^j x.\lambda^i x.t\left\{(v)x^j x^i/(v)x^i\right\} & \textit{if } x^j \notin FV(t) \textit{ and } 0\le j<i\le n \end{array}}$$

Proposition 1. *For any $0 \le j < i \le n$, $\mu^{i/j}$ can be derived from $fst^{i/j}$ and β^j.*

Definition 11. *We consider the following subsystems of Λ^n_η-reduction:*

- *$\boldsymbol{\beta}$ (resp. $\boldsymbol{\eta}$) is the subsystem of reductions $(\beta^i)_{0\leq i\leq n}$ (resp. $(\eta^i)_{0\leq i\leq n}$);*
- ***fst** is the subsystem made of reductions $(fst^{i/j})_{0\leq j<i\leq n}$;*
- *β^0_{var} is the restriction of β^0 to redex where the argument is a level-0 variable;*
- *$\boldsymbol{\beta}_{var}$ is the subsystem made of reductions β^0_{var} and $(\beta^i)_{1\leq i\leq n}$.*

Example 1. Λ^0 and Λ^1 are respectively λ-calculus and $\Lambda\mu$-calculus.

We shall consider here an example in Λ^i which is a CBN correspondent to the level i `Shift` of the CPS-hierarchy $\mathcal{S} = \lambda^0 x.\lambda^i y.(x^0)\lambda^0 z.(z^0)y^i$.

Consider $\mathcal{C}^{<i} = []ut_{1,1}\dots t_{1,n_1}x_1^{j_1}\dots t_{k,1}\dots t_{k,n_k}x_k^{j_k}$ such that for all $l\leq k$, $j_l<i$, we have $\mathcal{C}^{<i}(\mathcal{S}) \longrightarrow^\star_{\Lambda^i} \lambda^i y.(u)\lambda^0 z.(z^0)t_{1,1}\dots t_{1,n_1}x_1^{j_1}\dots t_{k,1}\dots t_{k,n_k}x_k^{j_k}y^i$ that is $\mathcal{S}$ stores any context of level strictly less than i in a continuation that can later be manipulated (for instance it can be composed with itself if $u = \lambda^0 x.(u')\lambda^0 y.(x^0)(x^0)y^0$). The flow of control is given to u only once an argument of level i (or higher) is reached, in which case $\lambda^i y$ is destroyed.

3.3 Meta-theory of the Stream Hierarchy

In this section, we state two essential theorems of Λ^n-calculi: confluence and separation. More details can be found in [30].

Confluence theorem. Confluence holds on closed terms. Such a restriction is necessary: $(\lambda^2 y.x)z^2$ reduces to x and to $(\lambda^0 y.\lambda^1 y'.\lambda^2 y''.x)z^2$ which cannot reduce to the same term.

Theorem 6. *For any $n\in\omega$ and any $t,u,v\in\Sigma^c_{\Lambda^n}$, if $t\longrightarrow^\star_{\Lambda^n_\eta} u,v$ then there exists $w\in\Sigma^c_{\Lambda^n}$ such that $u,v\longrightarrow^\star_{\Lambda^n_\eta} w$.*

As a corollary, Λ^j is a conservative extension of Λ^i, for any $i<j$:

Corollary 1. *Let $i<j\in\omega$ and $t,u\in\Sigma^c_{\Lambda^i}$. Then $t=_{\Lambda^i_\eta} u$ iff $t=_{\Lambda^j_\eta} u$.*

Böhm theorem. To state the separation theorem (*aka* Böhm theorem) for the stream hierarchy, we first define *canonical normal forms* for the hierarchy using the notion of *pre-redex*.

Definition 12. *$t\in\Sigma_{\Lambda^n}$ is a* **pre-redex** *if it is of the form $(\lambda^i x.t)y^j$ or $(\lambda^i x.t)u$ for $0\leq i,j\leq n$.*

Canonical normal forms (Λ^n-CNF) can be considered as those terms containing only **fst**-redexes such that a **fst**-reduction does not create any redex other than **fst**-redexes:

Definition 13. *A* **Λ^n-CNF** *is a $\boldsymbol{\beta\eta}$-normal form with no pre-redex.*

We can now state the separation result:

Theorem 7. *Let $n\in\omega$, $t,u\in\Sigma^c_{\Lambda^n}$. If t,u are non **fst**-equivalent Λ^n-CNF then there exists a context $\mathcal{C}[]$ st. $\mathcal{C}[t]\longrightarrow^\star_{\Lambda^n_\eta}\lambda^0 x,y.x^0$ and $\mathcal{C}[u]\longrightarrow^\star_{\Lambda^n_\eta}\lambda^0 x,y.y^0$.*

4 Translating the Stream Hierarchy into λ-Calculus

We define in this section sound and complete translations of the stream hierarchy into λ-calculus with pairs. These translations are inspired by the recent CPS translation for $\lambda\mu\widehat{\mathsf{tp}}$-calculus by Herbelin and Ghilezan [16]. Several translations into λ-calculus have been proposed for λμ-calculus in the literature. de Groote [10] was the first to study CPS translations for λμ-calculus. Lafont, Reus and Streicher [24] proposed a CPS translation for λ-calculus into λ-calculus with pairs which later led to a continuation semantics for λμ-calculus[36] and is very much related to CPS translations for λμ-calculus by Fujita [14] or Lassen [25]. A by-product of this section is to provide a sound and complete CPS translation for Λμ-calculus. We recall the definition of the λ-calculus with pairs.

Definition 14. *Terms of* **λ-calculus with pairs** *are given by the following syntax:*

$$\boxed{\Sigma_{\lambda\pi} \qquad t, u ::= x \mid \lambda x.t \mid (t)u \mid \langle t, u\rangle \mid (\pi_1)t \mid (\pi_2)t}$$

Definition 15. *Equations of λπ are βη (equationally) plus the following:*

$$\boxed{(\pi_1)\langle t_1, t_2\rangle =_{\pi_1} t_1 \qquad (\pi_2)\langle t_1, t_2\rangle =_{\pi_2} t_2 \qquad \langle (\pi_1)t, (\pi_2)t\rangle =_{SP} t}$$

Definition 16. *We assume that the set of variables of λ-calculus with pairs is $\mathcal{V} = \{k\} \uplus \mathcal{V}_0 \uplus \cdots \uplus \mathcal{V}_n$ and we define a translation $[-] : \Sigma_{\Lambda^n} \longrightarrow \Sigma_{\lambda\pi}$ as follows:*

$$\begin{aligned}
\left[x^0\right] &= \lambda k.(x^0)k \\
\left[\lambda^i x.t\right] &= \lambda k.((\lambda x^i.\,[t])(\pi_1)^{n-i+1}k)\langle \ldots \langle (\pi_2)(\pi_1)^{n-i}k, (\pi_2)(\pi_1)^{n-i-1}k\rangle \ldots, (\pi_2)k\rangle \\
\left[(t)x^i\right] &= \lambda k.([t])\langle \ldots \langle\langle x^i, (\pi_1)^{n-i}k\rangle, (\pi_2)(\pi_1)^{n-i-1}k\rangle \ldots, (\pi_2)k\rangle \\
[(t)u] &= \lambda k.([t])\langle \ldots \langle\langle [u], (\pi_1)^n k\rangle, (\pi_2)(\pi_1)^{n-1}k\rangle \ldots, (\pi_2)k\rangle
\end{aligned}$$

with $0 \le i \le n$ for $\left[\lambda^i x.t\right]$ and $0 < i \le n$ for $\left[(t)x^i\right]$.

Remark 8. *In the previous definition, we abbreviated $(\pi_i)(\pi_i)\ldots(\pi_i)t$ as $(\pi_i)^n t$.*

The definition for $[(t)u]$ when $u = x_0$ corresponds to instantiating the definition for $\left[(t)x^i\right]$ with $i = 0$. An alternative definition for $[(t)u]$ is thus possible: $[(t)u] = \left[(t)x^0\right]\left\{[u]/x^0\right\}$ if $x^0 \notin FV(t)$, if clause for $\left[(t)x^i\right]$ is extended to $i = 0$.

Example 2. Consider $t = \lambda^1 y_0 \ldots y_n.(x^0)t_1 \ldots t_m \in \Sigma_{\Lambda^1}$. Then one has:

$$[t] \longrightarrow^\star$$
$$\lambda k.(x^0)\langle\langle [t_1], \ldots \langle [t_m], (\pi_1)(\pi_2)^{m+1}k\rangle\rangle, (\pi_2)^{m+2}k\rangle \left\{(\pi_1)(\pi_2)^i k/y_i^1,\ 0 \le i \le m\right\}$$

The translation is sound and complete with respect to Λ^n_η-equational theory:

Theorem 9. *For any $n \in \omega$, $t, u \in \Sigma^c_{\Lambda^n}$, $t =_{\Lambda^n_\eta} u$ iff $[t] =_{\beta\eta\pi SP} [u]$.*

For the completeness part, we study the image of Σ_{Λ^n} terms by the translation which is characterized by the terms T defined by the following grammar:

Definition 17. *The target of the CPS can be defined by the following grammar:*

$$\begin{array}{lllll} T, K_0 & ::= x^0 \mid \lambda k.(T)K_{n+1} & \mid (\lambda x^i.T)K_i & \mid (\pi_1)K_1 & \textit{(for } 0 \leq i \leq n\textit{)} \\ K_i & ::= x^i \mid \langle K_{i-1}, K_i\rangle & \mid (\pi_2)K_i & \mid (\pi_1)K_{i+1} & \textit{(for } 0 < i \leq n\textit{)} \\ K_{n+1} & ::= k \quad \mid \langle K_n, K_{n+1}\rangle & \mid (\pi_2)K_{n+1} & & \end{array}$$

Proof. We only sketch the proof, more details are available in appendix and in the long version. Soundness is obtained by induction on the length of a proof of equality between t and u. Completeness is more involved. It mainly amounts to the following arguments:

- an inverse translation, $\_^\sharp$, is defined from the target language of Λ^n to Λ^{n+1};
- one proves that the inverse translation preserves equality *in* Λ^{n+1}, and thus: if $[t] =_{\beta\eta\pi SP} [u]$, then $[t]^\sharp =_{\Lambda_\eta^{n+1}} [u]^\sharp$;
- one then shows that $[t]^\sharp =_{\Lambda_\eta^{n+1}} t$ so that we can deduce that $t =_{\Lambda_\eta^{n+1}} u$ and
- finally we conclude thanks to the fact that Λ_η^{n+1} is a conservative extension of Λ_η^n (corollary 1): $t =_{\Lambda_\eta^n} u$. □

Remark 10. *It shall be noted that the proof of completeness is greatly simplified by the use of the hierarchy in the sense that the inverse translation translates back to Λ^{n+1} and not to Λ^n. Indeed, it can take advantage of the regularity of the structure of the $n+1^{th}$ continuation used in the translation.*

A sound and complete CPS translation for $\Lambda\mu$-calculus, $[]^{\Lambda\mu}$, is obtained by instantiating the previous result with $n = 1$. We have the following corollary:

Corollary 2. *For any $t, u \in \Sigma^c_{\Lambda\mu}$, $t =_{\Lambda\mu} u$ if, and only if, $[t]^{\Lambda\mu} =_{\beta\eta\pi SP} [u]^{\Lambda\mu}$.*

5 An Operational Investigation of the Stream Hierarchy

In the final section of his seminal paper, Parigot outlined an abstract machine for $\lambda\mu$-calculus. Later, de Groote [11] and Streicher and Reus [36] studied abstract machines for $\lambda\mu$-calculus. We shall be interested in this section in abstract machines for the Stream hierarchy. We shall now define abstract machines which compute Λ^n-head normal forms. In the following, we do not consider extensionality rules which are not necessary to compute head normal forms.

Definition 18. *Λ^n-head normal forms are defined by the following grammar:*

$$\boxed{\begin{array}{lll} h ::= g & \mid \lambda^i x.h & \textit{for } 0 \leq i \leq n \\ g ::= x^0 & \mid (g)t \quad \mid (g)x^i & \textit{for } 0 < i \leq n,\ t \textit{ denotes an arbitrary } \Lambda^n\textit{-term} \end{array}}$$

In the next definition we introduce constants representing variables in order to compute head normal forms (and not only weak head normal forms).

Definition 19. *Let $0 \leq i \leq n$. Constants of level i are defined as*

$$\boxed{\mathfrak{c}^i = x^i_{\rho_i} \mid v(\mathfrak{c}^j)^i_{\rho_i}}$$

with $x \in \mathcal{V}$, $i < j$ and ρ_i a finite sequence of integers $k, 0 \leq k < i$ (ϵ denotes the empty sequence). We shall also consider particular constants which shall represent empty contexts: $\bot_i$, $0 \leq i \leq n{+}1$. The structure of constants takes into account the need for treating the case of fst-rules. For instance x^j_ϵ will represent the variable x^j in the left-hand side of $\lambda^j x.t \longrightarrow_{fst^{j/i}} \lambda^i y.\lambda^j x'.t'$, while $v(x^j_\epsilon)^i_\epsilon$ and $x^j_{[i]}$ will respectively represent variable y^i and x'^j in the right-hand side.

Moreover, the following notions are needed to define the machine:

Definition 20. *We define by mutual recursion* **contexts of level** i*, $0 \leq i \leq n+1$,* **closures** *and* **environments***:*

- *a closure is a pair of a term t and an environment e, denoted $t[e]$;*
- *an environment e is a partial function which, when defined, associates to a variable of level i a context of level i;*
- *a context S_0 of level 0 is defined as follows: $S_0 ::= \bot_0 \mid \mathfrak{c}^0 \mid u[e]$;*
- *a context S_i of level i ($i \geq 1$) is defined as follows: $S_i ::= \bot_i \mid \mathfrak{c}^i \mid S_{i-1} \cdot S_i$.*

We set $\bot_i \cdot S_{i+1}$ to be equal to S_{i+1}, and $(S^1_i \cdot \ldots (S^n_i \cdot \bot_{i+1})) \cdot (S_{i+1} \cdot S_{i+2})$ to $(S^1_i \cdot \ldots (S^n_i \cdot S_{i+1})) \cdot S_{i+2}$. These equalities allow us to assume that if S is of the form $(((S^1_i \cdot \ldots (S^n_i \cdot \bot_{i+1})) \cdot S_{i+2}) \ldots S_k)$, then either $\forall j, i+2 \leq j \leq k, S_j = \bot_j$ or it is of the form $((((((S^1_i \cdot \ldots (S^n_i \cdot \bot_{i+1})) \cdot \bot_{i+2}) \ldots \bot_{j-1}) \cdot \mathfrak{c}^j_\rho) \cdot S_{j+1}) \ldots S_k)$.

Definition 21. *We define $\boldsymbol{pop}^i(S_{n+1})$ and $\boldsymbol{push}(S_i, S_{n+1})$ as follows:*

- *$\boldsymbol{push}(S_i, S_j)$ (with $i < j$):*
  - *$\boldsymbol{push}(\bot_i, S_j) = S_j$;*
  - *$\boldsymbol{push}(S_i, \bot_j) = ((S_i \cdot \bot_{i+1}) \cdot \cdots \cdot \bot_j)$ if $S_i \neq \bot_i$;*
  - *$\boldsymbol{push}(S_i, \mathfrak{c}^j_\rho) = ((S_i \cdot \bot_{i+1}) \cdot \cdots \cdot \bot_{j-1}) \cdot \mathfrak{c}^j_\rho$ if $S_i \neq \bot_i$;*
  - *$\boldsymbol{push}(S_i, S_{i+1}) = (S_i \cdot S_{i+1})$ if $S_i \neq \bot_i$;*
  - *$\boldsymbol{push}(S_i, S_j \cdot S_{j+1}) = (\boldsymbol{push}(S_i, S_j) \cdot S_{j+1})$ if $S_i \neq \bot_i$.*
- *$\boldsymbol{pop}^i(S_{n+1}) = \boldsymbol{pop}^{i,n+1}(S_{n+1})$ with $\boldsymbol{pop}^{i,j}(S_j)$ (for $i < j$):*
  - *$\boldsymbol{pop}^{i,j}(\bot_j) = (\bot_i, \bot_j)$;*
  - *$\boldsymbol{pop}^{i,j}(\mathfrak{c}^j_\rho) = (v(\mathfrak{c}^j_\rho)^i_\epsilon, \mathfrak{c}^j_{\rho \cdot i})$;*
  - *$\boldsymbol{pop}^{i,j+1}(((S^1_{i-1} \cdot \ldots S^n_{i-1} \cdot \bot_i) \cdot \bot_{i+1}) \cdot \ldots \bot_{j+1}) = (S^1_{i-1} \cdot \ldots S^n_{i-1} \cdot \bot_i, \bot_{j+1})$;*
  - *$\boldsymbol{pop}^{i,j+1}(((((S^1_{i-1} \cdot \ldots S^n_{i-1} \cdot \bot_i) \cdot \cdots \cdot \bot_{k-1}) \cdot \mathfrak{c}^k_\rho) \cdot S_{k+1}) \cdot \cdots \cdot S_{j+1}) = (S^1_{i-1} \cdot \ldots S^n_{i-1} \cdot v(\mathfrak{c}^k_\rho)^i_\epsilon, ((\mathfrak{c}^k_{\rho \cdot i} \cdot S_{k+1}) \ldots S_{j+1}))$. Otherwise, one has:*
  - *$\boldsymbol{pop}^{i,j+1}(S_j \cdot S_{j+1}) = (S'_i, S'_{j+1})$ if $\boldsymbol{pop}^{i,j}(S_j) = (S'_i, S''_j)$ and $S'_{j+1} = \boldsymbol{push}(S''_j, S_{j+1})$.*

We now define the Λ^n-KAM:

Definition 22. States of Λ^n-KAM *have the form $\lambda^{i_1} {x_1}^{i_1}_\epsilon \ldots . \lambda^{i_n} {x_n}^{i_n}_\epsilon . \langle t, [e], S_{n+1} \rangle$ where $t \in \Sigma_{\Lambda^n}$, e is an environment and S_{n+1} is a context of level $n+1$. States are abbreviated as $\overrightarrow{\lambda} \langle t[e]\ S_{n+1} \rangle$ when the prefix of abstractions is irrelevant.*

An **initial state** *of Λ^n-KAM is of the form $\langle t, [\emptyset], \bot_{n+1} \rangle$.*

Definition 23. *The transitions of the machines are the following:*

$$\begin{array}{llllll}
\overrightarrow{\lambda}\langle x^0 & [e]\ S\rangle \longrightarrow \overrightarrow{\lambda}\langle t & [e'] & S\rangle & \textit{if } e(x^0) = t[e'] \\
\overrightarrow{\lambda}\langle (t)u & [e]\ S\rangle \longrightarrow \overrightarrow{\lambda}\langle t & [e] & S'\rangle & \textit{with } S' = \textbf{\textit{push}}(u[e], S) \\
\overrightarrow{\lambda}\langle (t)x^i & [e]\ S\rangle \longrightarrow \overrightarrow{\lambda}\langle t & [e] & S'\rangle & \textit{with } S' = \textbf{\textit{push}}(e(x^i), S) \\
\overrightarrow{\lambda}\langle \lambda^i x.t & [e]\ S\rangle \longrightarrow \overrightarrow{\lambda}\langle t & [e'] & S'\rangle & \textit{if } \textbf{\textit{pop}}^i(S) = (S'_i, S'),\ e' = [e, x^i = S'_i] \\
 & & & & \textit{and } S'_i \neq S^1_{i-1} \cdot \ldots (S^n_{i-1} \cdot \bot_i) \\
\overrightarrow{\lambda}\langle \lambda^i x.t & [e]\ S\rangle \longrightarrow \overrightarrow{\lambda}\lambda^i x^i_\epsilon.\langle t & [e'] & \bot_{n+1}\rangle & \textit{if } \textbf{\textit{pop}}^i(S) = (S^1_{i-1} \cdot \ldots (S^n_{i-1} \cdot \bot_i), \bot_{n+1}) \\
 & & & & \textit{and } e' = [e, x^i = S^1_{i-1} \cdot \ldots (S^n_{i-1} \cdot x^i_\epsilon)]
\end{array}$$

The only case when the machine cannot reduce is when the machine state is in case $\lambda^{i_1} x^{i_1}_{1\epsilon}. \ldots \lambda^{i_n} x^{i_n}_{n\epsilon}.\langle x^0, [e], S\rangle$ and x^0 is associated by e to a variable constant of level 0, $\mathfrak{c}^0$, and not to a closure $t[e']$ since there is no rule for reducing this case (it is easy to check that when the initial state is made of a closed term, this is indeed the only case which can stop the machine). The final states of the machine are thus of the form:

$$\boxed{\lambda^{i_1} x^{i_1}_{1\epsilon}. \ldots \lambda^{i_n} x^{i_n}_{n\epsilon}.\langle \mathfrak{c}^0, [e], S\rangle}$$

In that case, we have reached the head variable and obtained the head normal form, the prefix of $\lambda^i x^i_\epsilon$ which has been gathered during the computation is the prefix of abstractions of the head normal form (up to some **fst**-reduction which have been lazily performed in the term and shall be propagated during the reconstruction of the Λ^n-term). One actually has the following:

Theorem 11. *If t is a closed Λ^n-term, Λ^n-KAM stops after a computation from initial state $\langle t[\emptyset], \bot_{n+1}\rangle$ if and only if t has a head normal form.*

Moreover, from the constant of level 0 which is the left-component of the final state, one can compute the head variable of the head normal form and recursively the complete head normal form.

6 Relating the Stream Hierarchy and the CPS Hierarchy

The aim of this section is to make clear how the Stream hierarchy relates to Danvy & Filinski's CPS hierarchy and to actually show that the Stream hierarchy is indeed a call-by-name analogous to the CPS hierarchy, that is a CBN hierarchy of delimited continuations. For this purpose, we first introduce a new hierarchy of calculi, the $\lambda\mu\widehat{\mathsf{tp}}_n$-calculi that we use as mediators between the two CBN/CBV hierarchies, following a method recently developed by Herbelin and Ghilezan.

6.1 $\lambda\mu\widehat{\mathsf{tp}}_n$-calculi

Definition 24 ($\lambda\mu\widehat{\mathsf{tp}}_n$-calculi). *Let $n \in \omega$. $\lambda\mu\widehat{\mathsf{tp}}_n$*-**terms** *$(t, u, v, \dots \in \Sigma_{\lambda\mu\widehat{\mathsf{tp}}_n})$ are defined by the following syntax (with $q ::= \alpha \mid \widehat{\mathsf{tp}}$):*

$$\boxed{\Sigma_{\lambda\mu\widehat{\mathsf{tp}}_n} \qquad t, u ::= x \mid \lambda x.t \mid (t)u \mid \mu^i q.c_i \qquad c_i ::= [q^i]t \quad (1 \leq i \leq n)}$$

CBV and CBN $\lambda\mu\widehat{\mathsf{tp}}_n$-calculi can be naturally considered. In the CBV case, values and evaluation contexts are defined as $V ::= x \mid \lambda x.t$ and $E^i_v ::= [] \mid (E^i_v)t \mid (V)E^i_v \mid \mu^j\widehat{\mathsf{tp}}.[q^j]E^i_v$, $1 \leq j < i$ while in the CBN case, every term is a value and evaluation contexts are $E^i ::= [] \mid (E^i)t \mid \mu^j\widehat{\mathsf{tp}}.[q^j]E^i$, $1 \leq j < i$.

Definition 25 (CBN/CBV $\lambda\mu\widehat{\mathsf{tp}}_n$equational theories)
CBV $\lambda\mu\widehat{\mathsf{tp}}_n$ equational theory (written $=_{\lambda\mu\widehat{\mathsf{tp}}_n^{cbv}}$) is defined by the following rules:

(β_v)	$(\lambda x.t)V = t\{V/x\}$		(η_v)	$\lambda x.(V)x = V$	*if* $x \notin FV(V)$
$(\eta^i_{\widehat{\mathsf{tp}}v})$	$\mu^i\widehat{\mathsf{tp}}.\left[\widehat{\mathsf{tp}}^i\right]V = V$		(η^i_μ)	$\mu^i\alpha.[\alpha^i]t = t$	*if* $\alpha^i \notin FV(t)$
$(\mu^i_{\widehat{\mathsf{tp}}})$	$\left[\widehat{\mathsf{tp}}^i\right]\mu^i\widehat{\mathsf{tp}}.c_i = c_i$		(μ^i_v)	$[q^i]E^{i-1}_v[\mu^i\alpha.c_i] = c_i\left\{[q^i]E^{i-1}_v[u]/[\alpha^i]u\right\}$	
(β^i_Ω)	$(\lambda x.E^i_v[x])\mu^i\widehat{\mathsf{tp}}.c_i = E^i_v[\mu^i\widehat{\mathsf{tp}}.c_i]$				
$(\mu'^i_{\widehat{\mathsf{tp}}})$	$\left[\widehat{\mathsf{tp}}^l\right]\mu^i\alpha.c_i = \left[\widehat{\mathsf{tp}}^l\right]\mu^i\widehat{\mathsf{tp}}.ci\left\{\widehat{\mathsf{tp}}^i/\alpha^i\right\}$	$(i \leq l)$			
(μ^i_{let})	$\mu^j\alpha.[q^j](\lambda x.t)\mu^i\widehat{\mathsf{tp}}.c_i = (\lambda x.\mu^j\alpha.[q^j]t)\mu^i\widehat{\mathsf{tp}}.c_i$	$(j \leq i+1)$			

$=_{\lambda\mu\widehat{\mathsf{tp}}_n^{cbn}}$ is defined by keeping only rules $\beta_v, \eta_v, \mu^i_v, \mu^i_{\widehat{\mathsf{tp}}}, \eta^i_{\widehat{\mathsf{tp}}v}$ and η^i_μ, by considering every terms as a value, E^i as evaluation contexts and constraining q^i to be α^i. CBN rules are denoted by droping the v subscripts in the rule names.

Definition 26 (Translations between $\lambda\mathcal{S}_n$ and $\lambda\mu\widehat{\mathsf{tp}}_n$)

$\lvert\langle t\rangle_i\rvert^{\mathcal{S}\rangle\widehat{\mathsf{tp}}} = \mu^i\widehat{\mathsf{tp}}.[\widehat{\mathsf{tp}}^i]\lvert t\rvert^{\mathcal{S}\rangle\widehat{\mathsf{tp}}}$	$\lvert\mathcal{S}_i k.t\rvert^{\mathcal{S}\rangle\widehat{\mathsf{tp}}} = \mu^i\alpha.[\widehat{\mathsf{tp}}^i](\lambda k.\lvert t\rvert^{\mathcal{S}\rangle\widehat{\mathsf{tp}}})\lambda x.\mu^i\widehat{\mathsf{tp}}.[\alpha^i]x$
$\lvert\mu^i\widehat{\mathsf{tp}}.c_i\rvert^{\widehat{\mathsf{tp}}\rangle\mathcal{S}} = \langle\lvert c_i\rvert^{\widehat{\mathsf{tp}}\rangle\mathcal{S}}\rangle_i$	$\lvert\mu^i\alpha.c_i\rvert^{\widehat{\mathsf{tp}}\rangle\mathcal{S}} = \mathcal{S}_i k^i_\alpha.\lvert c_i\rvert^{\widehat{\mathsf{tp}}\rangle\mathcal{S}}$
$\lvert[\widehat{\mathsf{tp}}^i]t\rvert^{\widehat{\mathsf{tp}}\rangle\mathcal{S}} = \lvert t\rvert^{\widehat{\mathsf{tp}}\rangle\mathcal{S}}$	$\lvert[\alpha^i]q\rvert^{\widehat{\mathsf{tp}}\rangle\mathcal{S}} = (k^i_\alpha)\lvert t\rvert^{\widehat{\mathsf{tp}}\rangle\mathcal{S}}$

Definition 27 (Translations between Λ^n and $\lambda\mu\widehat{\mathsf{tp}}_n$)

$\lvert\lambda^i x.t\rvert^{\Lambda\rangle\widehat{\mathsf{tp}}} = \mu^i\alpha_x.[\widehat{\mathsf{tp}}^i]\lvert t\rvert^{\Lambda\rangle\widehat{\mathsf{tp}}}$	$\lvert\mu^i\widehat{\mathsf{tp}}.c_i\rvert^{\widehat{\mathsf{tp}}\rangle\Lambda} = \lvert c_i\rvert^{\widehat{\mathsf{tp}}\rangle\Lambda}$	$\lvert\mu^i\alpha.c_i\rvert^{\widehat{\mathsf{tp}}\rangle\Lambda} = \lambda^i x_\alpha.\lvert c_i\rvert^{\widehat{\mathsf{tp}}\rangle\Lambda}$
$\lvert(t)x^i\rvert^{\Lambda\rangle\widehat{\mathsf{tp}}} = \mu^i\widehat{\mathsf{tp}}.[\alpha^i_x]\lvert t\rvert^{\Lambda\rangle\widehat{\mathsf{tp}}}$	$\lvert[\widehat{\mathsf{tp}}^i]t\rvert^{\widehat{\mathsf{tp}}\rangle\Lambda} = \lvert t\rvert^{\widehat{\mathsf{tp}}\rangle\Lambda}$	$\lvert[\alpha^i]t\rvert^{\widehat{\mathsf{tp}}\rangle\Lambda} = (\lvert t\rvert^{\widehat{\mathsf{tp}}\rangle\Lambda})x^i_\alpha$

Theorem 12. *For any $n \in \omega$, Λ^n is in eq. correspondence with CBN $\lambda\mu\widehat{\mathsf{tp}}_n$:*
let $t, u \in \Sigma^c_{\Lambda^n}$, $t =_{\Lambda^n_\eta} u \Rightarrow \lvert t\rvert^{\Lambda\rangle\widehat{\mathsf{tp}}} =_{\lambda\mu\widehat{\mathsf{tp}}_n^{cbn}} \lvert u\rvert^{\Lambda\rangle\widehat{\mathsf{tp}}}$
let $t, u \in \Sigma^c_{\lambda\mu\widehat{\mathsf{tp}}_n}$, $t =_{\lambda\mu\widehat{\mathsf{tp}}_n^{cbn}} u \Rightarrow \lvert t\rvert^{\widehat{\mathsf{tp}}\rangle\Lambda} =_{\Lambda^n_\eta} \lvert u\rvert^{\widehat{\mathsf{tp}}\rangle\Lambda}$

In order to study the correspondence with CPS hierarchy, we recall Kameyama's axiomatization of $\lambda\mathcal{S}_n$ [19]:

Definition 28. *$=_{\lambda\mathcal{S}_n}$ is defined as:*

(β_v)	$(\lambda x.t)V$	$= t\{V/x\}$	
(η_v)	$\lambda x.(V)x$	$= V$	*if* $x \notin FV(V)$
(β_Ω)	$(\lambda x.E^0_v[x])t$	$= E^0_v[t]$	*if* $x \notin FV(E^0_v)$
$(Reset - Value)$	$\langle V\rangle_i$	$= V$	
$(Reset - lift)$	$\langle(\lambda x.t)\langle u\rangle_i\rangle_j$	$= (\lambda x.\langle t\rangle_j)\langle u\rangle_i$	$j \leq i$
$(\mathcal{S} - reset)$	$\mathcal{S}_i k.\langle t\rangle_i$	$= \mathcal{S}_i k.t$	
$(\mathcal{S} - elim)$	$\mathcal{S}_i k.(k)\langle t\rangle_{i-1}$	$= \langle t\rangle_{i-1}$	$k \notin FV(t)$
$(\mathcal{S} - lift)$	$\langle E^{j-1}_v[\mathcal{S}_j k.t]\rangle_i$	$= \langle t\left\{\lambda x.\langle E^{j-1}_v[x]\rangle_j/k\right\}\rangle_i$	$x \notin FV(kE^{j-1}_v)$

Theorem 13. *For any $n \in \omega$, CBV $\lambda\mu\widehat{\mathsf{tp}}_n$ simulates $\lambda\mathcal{S}_n$: let $t, u \in \Sigma^{c}_{\lambda\mathcal{S}_n}$, $t =_{\lambda\mathcal{S}_n} u \Rightarrow |t|^{\mathcal{S}\rangle\widehat{\mathsf{tp}}} =_{\lambda\mu\widehat{\mathsf{tp}}_n^{cbv}} |u|^{\mathcal{S}\rangle\widehat{\mathsf{tp}}}$.*

Remark 14. *If we have only implication $\Rightarrow$ and not the converse, it is solely because $\lambda\mu\widehat{\mathsf{tp}}_n$ makes use of structural substitution and thus that some reductions are anticipated in $\lambda\mu\widehat{\mathsf{tp}}_n$ compared to the reduction in $\lambda\mathcal{S}_n$. This already occurs at the first level of the hierarchy [16] and is analyzed in [2].*

7 Conclusion

In this paper we introduced a new hierarchy of calculi, the $(\Lambda^n)_{n\in\omega}$-calculi, called the *stream hierarchy*. This hierarchy generalizes both λ-calculus and $\Lambda\mu$-calculus. $(\Lambda^n)_{n\in\omega}$-calculi have layered, or hierarchical, abstractions as well as variables with levels and its reduction system naturally extends the one for $\Lambda\mu$-calculus. The main related works are the CBV studies of delimited continuations and of the CPS hierarchiy and most notably works by Danvy, Filinski, Hasegawa and Kameyama [5,8,13,18,19] and the works on CBN delimited control by Ghilezan, Herbelin and Kiselyov [16,21]. The main results of the paper are:

– Church-Rosser and Böhm theorem for the hierarchy which ensures that the hierarchy is well-structured;
– sound and complete CPS translations for the hierarchy. The completeness proof strongly relies on conservativity results between different layers of the hierarchy allowing for simpler completeness proofs compared to more traditional translations as Fujita's CPS adapted to $\Lambda\mu$-calculus;
– an operational semantics for the hierarchy obtained by constructing abstract machines, the Λ^n-KAM, inspired from Krivine abstract machine for λ-calculus. The Λ^n-KAMs compute Λ^n-head normal forms;
– finally, we established that the stream hierarchy is indeed a hierarchy of delimited continuations in call-by-name, by mediating between the CPS hierarchy and the stream hierarchy thanks to the $\lambda\mu\widehat{\mathsf{tp}}_n$-calculi.

As a conclusion, we have developed a(n almost) complete study of the stream hierarchy. Our contribution evidences that the Stream hierarchy is a CBN hierarchy of delimited continuations and that fruitful connections exist between delimited control and infinitary calculi which underly $\Lambda\mu$-calculus and the entire stream hierarchy. However, some more developments are still to be done, which are left for future work:

– the CPS translations for the hierarchy can be used for a semantical study of the hierarchy. However, we are also interested in developping Böhm tree semantics for $\Lambda\mu$-calculus and the stream hierarchy;
– the CPS translations and the abstract machines considered in this paper have many similarities. It would be of interest to study how the abstract machines can be generated from the CPS semantics;
– the Λ^n-KAM has a structure (states and reductions) very similar to abstract machines for the CPS hierarchy [8,5]. We shall make this relation clear;

– we developed an untyped study of the stream hierarchy but a typed study of the hierarchy would also be of interest;
– the stream hierarchy that we considered here is indexed by ω. However, it can straightforwardly be made more general by indexing the hierarchy by a larger ordinal while preserving most results. We limited our presentation to ω for two reasons: for simplicity, first, but also because the CPS hierarchy is itself limited to ω. We conjecture that the CPS hierarchy can as well be extended above ω which could actually be interesting for several applications of the hierarchy where it might be of interest to have a delimiter that can delimit an infinite number of different `shift` operators;
– the Stream interpretation of $\Lambda\mu$-calculus and the links with infinitary calculi have been very influential. We shall develop these directions in future works. See [35] for some early developments.

Finally, we think that the ability to develop the stream hierarchy as a natural generalization of $\Lambda\mu$-calculus is a hint of the fact that $\Lambda\mu$-calculus is a calculus with a strong structure: this hierarchical extension could not have been developed based on Parigot's syntax for instance (but for adding a dynamically bound variable as we did with $\lambda\mu\widehat{\mathtt{tp}}_n$-calculi).

Acknowledgments. The author wishes to thank Hugo Herbelin, Luca Paolini, Mauro Piccolo, Kazushige Terui and Simona Ronchi della Rocca for helpful discussions and fruitful comments as well as the anonymous reviewers.

References

1. Ariola, Z., Herbelin, H.: Minimal classical logic and control operators. In: Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J. (eds.) ICALP 2003. LNCS, vol. 2719. Springer, Heidelberg (2003)
2. Ariola, Z., Herbelin, H.: Control Reduction Theories: the Benefit of Structural Substitution. In: JFP. Includes a Historical Note by Matthias Felleisen (2007)
3. Berarducci, A.: Infinite λ-calculus and non-sensible models. In: Logic and Algebra 1994. Lect. Notes in Pure and App. Math. Series, vol. 180. Marcel Dekker, New York (1996)
4. Berarducci, A., Dezani, M.: Infinite λ-calculus and types. TCS, 212 (1999)
5. Biernacka, M., Biernacki, D., Danvy, O.: An operational foundation for delimited continuations in the CPS hierarchy. Logical Meth. in Comp. Science 1(2) (2005)
6. Böhm, C.: Alcune proprietà delle forme $\beta\eta$-normali nel λK-calcolo. Publicazioni dell'Istituto per le Applicazioni del Calcolo 696 (1968)
7. Danvy, O., Filinski, A.: Abstracting control. In: LISP and Funct. Prog. (1990)
8. Danvy, O., Yang, Z.: An operational investigation of the CPS hierarchy. In: Swierstra, S.D. (ed.) ESOP 1999. LNCS, vol. 1576, pp. 224–242. Springer, Heidelberg (1999)
9. David, R., Py, W.: $\lambda\mu$-calculus and Böhm's theorem. J. of Symb. Logic (2001)
10. de Groote, P.: A CPS-translation of the $\lambda\mu$-calculus. In: Tison, S. (ed.) CAAP 1994. LNCS, vol. 787, pp. 85–99. Springer, Heidelberg (1994)
11. de Groote, P.: An environment machine for the λ-calculus. MSCS 8 (1998)

12. Felleisen, M., Friedman, D.P., Kohlbecker, E.E., Duba, B.F.: A syntactic theory of sequential control. TCS 52, 205–237 (1987)
13. Filinski, A.: Representing monads. In: POPL 1994, pp. 446–457. ACM, New York (1994)
14. Fujita, K.: A sound and complete cps-translation for λ-calculus. In: TLCA (2003)
15. Griffin, T.: A formulae-as-types notion of control. In: POPL. IEEE, Los Alamitos (1990)
16. Herbelin, H., Ghilezan, S.: An approach to CBN delimited continuations. In: POPL. ACM Sigplan, New York (2008)
17. Howard, W.A.: The formulae-as-type notion of construction, 1969. In: Essays in Comb. Logic, λ-Calculus, and Formalism, pp. 479–490. Academic Press, London (1980)
18. Kameyama, Y., Hasegawa, M.: A Sound and Complete Axiomatization of Delimited Continuations. In: ICFP 2003, pp. 177–188. SIGPLAN Notices (2003)
19. Kameyama, Y.: Axioms for control operators in the cps hierarchy. In: HOSC (2007)
20. Kennaway, R., Klop, J.W., Sleep, M.R., de Vries, F.-J.: Infinitary lambda calculus. TCS 175(1), 93–125 (1997)
21. Kiselyov, O.: Call-by-name linguistic side effects. In: ESSLLI 2008 Workshop on Symmetric calculi and Ludics for the semantic interpretation (2008)
22. Krivine, J.-L.: Lambda-calculus, Types and Models. Ellis Horwood (1993)
23. Krivine, J.-L.: A call-by-name lambda-calculus machine. In: HOSC (2005)
24. Lafont, Y., Reus, B., Streicher, T.: Continuations semantics or expressing implication by negation. Tech. Rep. 9321, Ludwig-Maximilians-Universität (1993)
25. Lassen, S.: Head normal form bisimulation for pairs and the λ-calculus. In: Logic In Computer Science. IEEE Computer Society Press, Los Alamitos (2006)
26. Parigot, M.: Free deduction: An analysis of "computations" in classical logic. In: Voronkov, A. (ed.) RCLP 1990 and RCLP 1991. LNCS (LNAI), vol. 592, pp. 361–380. Springer, Heidelberg (1992)
27. Parigot, M.: $\lambda\mu$-calculus: an algorithmic interpretation of classical natural deduction. In: Voronkov, A. (ed.) LPAR 1992. LNCS, vol. 624. Springer, Heidelberg (1992)
28. Parigot, M.: Proofs of strong normalisation for second order classical natural deduction. Journal of Symbolic Logic 62(4), 1461–1479 (1997)
29. Py, W.: Confluence en $\lambda\mu$-calcul. PhD thesis, Université de Savoie (1998)
30. Saurin, A.: A hierarchy for delimited continuations in call-by-name. long version at, `http://www.pps.jussieu.fr/~saurin/Publi/LM_hierarchy_long.pdf`
31. Saurin, A.: Separation with streams in the $\Lambda\mu$-calculus. In: LICS. IEEE, Los Alamitos (2005)
32. Saurin, A.: On the relations between the syntactic theories of $\lambda\mu$-calculi. In: Kaminski, M., Martini, S. (eds.) CSL 2008. LNCS, vol. 5213, pp. 154–168. Springer, Heidelberg (2008)
33. Saurin, A.: Une étude logique du contrôle, appliquée à la programmation fonctionnelle et logique. PhD thesis, École Polytechnique (September 2008)
34. Saurin, A.: Typing streams in the $\Lambda\mu$-calculus. ACM ToCL (to appear)
35. Saurin, A.: Standardization and Böhm trees for $\Lambda\mu$-calculus. In: FLOPS 2010. LNCS, vol. 6009. Springer, Heidelberg (to appear, 2010)
36. Streicher, T., Reus, B.: Classical logic, continuation semantics and abstract machines. Journal of Functional Programming 8(6), 543–572 (1998)