$\lambda\mu$-calculus and $\Lambda\mu$-calculus: a Capital Difference

Hugo Herbelin, Alexis Saurin

$\lambda\mu$-calculus and $\Lambda\mu$-calculus: a Capital Difference

Hugo Herbelin [a] & Alexis Saurin [b]

[a] *INRIA. Address: PPS, Université Paris 7, Case 7014, 75205 Paris Cedex 13, France. E-mail: Hugo.Herbelin@inria.fr*

[b] *Università di Torino. Address: Corso Svizzera 185, I-10149 Torino, Italy. E-mail: Saurin@di.unito.it*

Abstract

Since Parigot designed the $\lambda\mu$-calculus to algorithmically interpret classical natural deduction, several variants of $\lambda\mu$-calculus have been proposed. Some of these variants derived from an alteration of the original syntax due to de Groote, leading in particular to the $\Lambda\mu$-calculus of the second author, a calculus truly different from $\lambda\mu$-calculus since, in the untyped case, it provides a Böhm separation theorem that the original calculus does not satisfy.

In addition to a survey of some aspects of the history of $\lambda\mu$-calculus, we study connections between Parigot's calculus and the modified syntax by means of an additional toplevel continuation. This analysis is twofold: first we relate $\lambda\mu$-calculus and $\Lambda\mu$-calculus in the typed case using $\lambda\mu\mathsf{tp}$-calculus, which contains a toplevel continuation *constant* tp, and then we use calculi of the family of $\lambda\mu\widehat{\mathsf{tp}}$-calculi, containing a toplevel continuation *variable* $\widehat{\mathsf{tp}}$, to study the untyped setting and in particular relate calculi in the modified syntax.

Key words: $\lambda\mu$-calculus, $\Lambda\mu$-calculus, classical logic, control operators, toplevel continuation

1 Introduction

The problem of understanding the computational content of classical logic has been an active and vivid topic for a long time: while the Curry-Howard correspondence evidenced the connections between intuitionistic proofs and typed λ-calculus, the question of whether classical proofs could be endowed with a computational meaning remained mysterious for years.

In the early 90's however, the topic progressed in an impressively fast way

as several proposals were being made almost simultaneously. Soon after Griffin [Gri90] revealed that classical logic could be used to compute using Felleisen *et al* $\lambda_{\mathcal{C}}$-calculus [FFKD86] by typing operator $\mathcal{C}$ of type $\neg\neg A \to A$, Parigot designed the $\lambda\mu$-calculus [Par92] to provide an algorithmic interpretation of classical natural deduction. $\lambda\mu$-calculus was based on Parigot's earlier study of free deduction [Par91] and is built as an extension of λ-calculus, $\lambda\mu$-terms following the syntax [1] :

$$\Sigma_{\lambda\mu} \quad \ni \quad M ::= x \mid \lambda x.M \mid (M)M \mid \mu\alpha.[\beta]M$$

While other formalisms have been proposed (Girard's LC [Gir91,Mur92] of which the underlying untyped calculus could be formalized or Barbanera and Berardi's symmetric λ-calculus [BB96] for instance), $\lambda\mu$-calculus became widely studied and several variants of the calculus have been considered in the literature. In particular, de Groote [dG94] introduced a slight variant of the syntax of $\lambda\mu$-terms when studying the relation between $\lambda\mu$-calculus and $\lambda_{\mathcal{C}}$-calculus. De Groote's syntax for $\lambda\mu$-calculus was:

$$\Sigma_{\Lambda\mu} \quad \ni \quad M ::= x \mid \lambda x.M \mid (M)M \mid \mu\alpha.M \mid [\alpha]M$$

Though this change might seem trivial, it actually leads to two different calculi, at least in the untyped case: indeed, whereas Böhm separation theorem fails for Parigot's syntax [DP01], the second author could recover this property by considering an extension of $\lambda\mu$-calculus on $\Sigma_{\Lambda\mu}$, $\Lambda\mu$-calculus, which is very close to de Groote's presentation of $\lambda\mu$-calculus.

The aim of this article is *(i)* to survey some aspects of the history of $\lambda\mu$-calculus that led to such a variety of calculi and *(ii)* to study the connections between Parigot's calculus and calculi based on de Groote's alternative syntax. We shall develop this study by considering an additional continuation tp (a *toplevel* continuation) previously introduced in several works by the first author [AH03,AHS07a,HG08]. Our investigation of the connections between $\lambda\mu$ and $\Lambda\mu$ will be twofold: first we relate $\lambda\mu$-calculus and $\Lambda\mu$-calculus in the typed case and then we study the relation in the untyped case and in particular we relate de Groote's $\lambda\mu\epsilon$-calculus [dG98] with a variant of $\lambda\mu\widehat{\mathsf{tp}}$-calculus.

[1] We adopt Krivine's convention regarding the notation for application construction. First, it is the function which is surrounded by parentheses in an application and not the full application. Secondly, application is left-associative so that for an n-ary application, only the head function needs parentheses around as e.g. in $(\lambda x.M)\,N\,P\,Q$ (see [Kri93] for details).

2 The development of $\lambda\mu$-calculus and $\Lambda\mu$-calculus

The purpose of this extended historical section is to review some of the milestones of the development of $\lambda\mu$-calculus. We think that, almost 20 years after the discovery by Griffin that control operators provided a computational meaning to classical logic and the introduction of $\lambda\mu$-calculus by Parigot, such an historical survey may be useful to put things in the context of their discovery.

2.1 The genesis of $\lambda\mu$-calculus (1992)

Parigot derived his $\lambda\mu$-calculus in 1991 from his previous work on free deduction [Par91], a hybrid logic generalising both natural deduction and sequent calculus. $\lambda\mu$-calculus was presented at the LPAR conference in July 1992 [Par92]. One year before, in January 1990 at the POPL conference, Griffin [Gri90] revealed that classical logic could be executed in practice and that an appropriate candidate for computing with classical logic was Felleisen *et al*'s $\lambda_{\mathcal{C}}$-calculus [FFKD86].

Two elements immediately stressed $\lambda\mu$-calculus as an original contribution:

- $\lambda\mu$ was explicitly thought as an untyped calculus relevant to denote proofs in classical natural deduction and
- the focus was implicitly put on the call-by-name variant even though alternative reduction rules, specific to call-by-value, were also considered.

Definition 1 ($\lambda\mu$-calculus [Par92]) *The syntax of $\lambda\mu$-calculus is:*

$$\begin{array}{rcll} \Sigma_{\lambda\mu} & \ni & M,N ::= x \mid \lambda x.M \mid (M)\,M \mid \mu\alpha.c & \textit{(unnamed terms)} \\ & & c ::= [\alpha]M & \textit{(named terms)} \end{array}$$

where x ranges over a set of λ-variables and α over a (disjoint) set of μ-variables, also called continuation variables. *$\lambda\mu$-calculus reduction rules are presented in figure 1 where $c\{[\alpha](P)N/[\alpha]P\}$ denotes the capture-avoiding substitution of every subterm $[\alpha]P$ in c by $[\alpha](P)N$.*

Names in the left column were considered by Parigot in [Par93] while R_1, R_2 and S_1 are respectively called *logical reduction*, *structural reduction* and *renaming* in [Par92]. Moreover, notice that the two-levels syntax of $\lambda\mu$-calculus forces a constraint on terms in $\Sigma_{\lambda\mu}$ that we refer to as the $\mu[]$ *constraint*: any term with prefix $\mu\alpha$ has actually $\mu\alpha.[\beta]$ as prefix for some variable β. Also, note that rule S_1 applies to named terms only and should be rephrased as $\mu\gamma.[\beta]\mu\alpha.c \to \mu\gamma.c\{\beta/\alpha\}$ to apply on $\lambda\mu$-terms.

$$
\begin{array}{lll}
(R_1)\ (\lambda x.M)\,N & \to M\,\{N/x\} & \\
(R_2)\ (\mu\alpha.c)\,N & \to \mu\beta.c\,\{[\beta](P)N/[\alpha]P\} & \text{if } \beta \text{ not free in } c, N \\
(S_1)\ [\beta]\mu\alpha.c & \to c\,\{\beta/\alpha\} & \\
(S_2)\ \mu\alpha.[\alpha]M & \to M & \text{if } \alpha \text{ not free in } M
\end{array}
$$

Figure 1. $\lambda\mu$-calculus reduction rules.

Comment 1 *Retrospectively, one can point out the fact that structural reduction and renaming are two facets of the same notion, namely of the binding of the surrounding evaluation context of an expression. This latter observation which is obvious when observing $\lambda\mu$-calculus from the point of view of the "duality of computation" [CH00] or from the stream viewpoint of $\Lambda\mu$-calculus [Sau05] was apparently unknown of Parigot when designing $\lambda\mu$.*

A posteriori, we think that the fundamental discovery in $\lambda\mu$, compared to $\lambda_{\mathcal{C}}$, is the notion of structural substitution *which moves pieces of evaluation contexts along continuation variables. The more fundamental nature of structural substitution compared to the $\lambda_{\mathcal{C}}$ substitution of evaluation contexts reified as ordinary functions (i.e. as functions of the form $\lambda x.E[x]$ for E a given evaluation context), is undoubtable*[2] *. As for the at-the-time unsuspected duality between substitution of terms along ordinary variables and substitution of evaluation contexts along continuation variables, it comes from Curien and Herbelin [CH00] (see Section 2.4).*

Based on the fact that continuation variables are used in $\lambda\mu$-calculus as placeholders for evaluation contexts, a more precise name for calling them could be "evaluation contexts variables". By convenience, we shall continue to call them "continuation variables" (as first suggested in [HS97]) even though the term "continuation" itself has no precise definition in this context.

In his LPAR paper, Parigot attempts to show how $\lambda\mu$ could be used to interpret the full strength of classical logic, namely to interpret $\neg\neg A \to A$. He adopts the convention that variables of type $\bot$ must be hidden in typing

[2] If E is some evaluation context, say $[\,]\,N$, the $\mathcal{C}$ operator of $\lambda_{\mathcal{C}}$ typically reduces $E[\mathcal{C}\,\lambda k.M]$ to $\mathcal{C}\,\lambda k'.M\,\{\lambda x.(k')\,E[x]/k\}]$ where we can see that E has been first *reified* into the regular function $\lambda x.(k')\,E[x]$ before being substituted in place of k. In a call-by-name setting, this further reduces to $\mathcal{C}\,\lambda k'.M\,\{(k')\,E[P]/(k)\,P\}]$ if ever all occurrences of k in M are applied to some term. However, in a call-by-value, there is no reason that $M\,\{\lambda x.(k')\,E[x]/k\}$ ever reduces to $M\,\{(k')\,E[P]/(k)\,P\}]$ since the P are not necessarily values. Conversely, $\mathcal{C}$ is directly definable with its intended semantics from μ by setting $\mathcal{C}\,M \triangleq \mu\alpha.[\mathsf{tp}](M)\,\lambda x.\mu\delta.[\beta]x$ (see Section 2.5 for explanations about tp and [AH07] for a detailed analysis of the correspondence between $\lambda_{\mathcal{C}}$ and $\lambda\mu$).

$$\frac{}{\Gamma, x : A \vdash x : A; \Delta} \qquad \frac{\Gamma, x : A \vdash M : B; \Delta}{\Gamma \vdash \lambda x.M : A \to B; \Delta} \qquad \frac{\Gamma \vdash M : A \to B; \Delta \qquad \Gamma \vdash N : A; \Delta}{\Gamma \vdash (M)N : B; \Delta}$$

$$\frac{\Gamma \vdash c; \Delta, \alpha : A}{\Gamma \vdash \mu\alpha.c : A; \Delta} \qquad \frac{\Gamma \vdash M : A; \Delta, \alpha : A}{\Gamma \vdash [\alpha]M; \Delta, \alpha : A}$$

Figure 2. Parigot's typing system for $\lambda\mu$-calculus.

derivations for classical logic (the full original typing system for Parigot's $\lambda\mu$-calculus is shown in figure 2). Nevertheless, this convention cannot hide the variables of type $\bot$ that occur in the terms themselves. Especially, the term that proves $\neg\neg A \to A$ in Parigot's $\lambda\mu$-calculus [3] is not a closed term, which remains unsatisfactory from the proof-theoretic point of view (one would expect of a proof-term that its free variables refer to proper assumptions).

Two lines of work provide more satisfactory answers to this question while still preserving the benefit of structural substitution: de Groote's extended syntax [dG94,dG98] and Ariola *et al*'s [AH03,AHS07a] introduction of an additional continuation constant `tp` as we shall see below.

2.2 De Groote's modified syntax for $\lambda\mu$-calculus (1994)

On the relations between $\lambda\mu$ and $\lambda_{\mathcal{C}}$. In 1993, de Groote studied some aspects of the relation between $\lambda\mu$-calculus and Felleisen *et al*'s $\lambda_{\mathcal{C}}$-calculus [FFKD86] (his work was presented at LPAR in 1994 [dG94]) which led him to introduce a syntactic variant of $\lambda\mu$-calculus and of its system of simple types.

$\lambda_{\mathcal{C}}$-calculus was initially motivated as a formal language for studying control constructs such as `call/cc` in Scheme [4] and de Groote's approach to $\lambda_{\mathcal{C}}$-calculus is rather ad hoc. Indeed, on the one hand, while Felleisen *et al* treat call-by-value only, de Groote focuses on a call-by-name formulation of $\lambda_{\mathcal{C}}$-calculus. On the other hand while Felleisen *et al* basically consider untyped $\lambda_{\mathcal{C}}$-calculus, de Groote focuses on a typed variant.

[3] Parigot shows $\aleph = \lambda y.\mu\alpha.[\beta](y)\lambda x.\mu\delta.[\alpha]x$ as a term of type $\neg\neg A \to A$, where β is a free continuation variable of type $\bot$.

[4] Even if $\lambda_{\mathcal{C}}$ actually turned out to be unable to faithfully simulate the operational semantics of `call/cc`, see [AH07].

$$
\begin{array}{lll}
(\beta) & (\lambda x.M)\, N & \rightarrow M\,\{N/x\} \\
(\mathcal{C}_L) & (\mathcal{C}\, M)\ N & \rightarrow \mathcal{C}\ \lambda k.(M)\, \lambda f.(k)\ (f)\ N \\
(\mathcal{C}_{top})\ \mathcal{C}\, M & & \rightarrow \mathcal{C}\ \lambda k.(M)\, \lambda f.(k)\ f
\end{array}
$$

Figure 3. de Groote's presentation of call-by-name $\lambda_{\mathcal{C}}$ reduction rules.

The syntax considered by de Groote for call-by-name $\lambda_{\mathcal{C}}$ was:

$$\Sigma_{\lambda\mathcal{C}} \quad \ni \quad M, N ::= x \mid \lambda x.M \mid (M)\, M \mid \mathcal{C}\, M$$

while the reduction rules are presented in figure 3. Some remarks deserve to be done:

- The rule (C_L) already appears in Felleisen *et al* but with the original form $(\mathcal{C}\, M)\ N \rightarrow \mathcal{C}\ \lambda k.(M)\, \lambda f.(\mathcal{A})\ (k)\ (f)\ N$ where $\mathcal{A}$, defined as $\lambda x.\mathcal{C}\ \lambda k.x$, is an *abort* operator that throws its evaluation context away. The removal of *abort* from the original $\mathcal{C}_L$ of Felleisen is a relevant simplification of the rule as it was also observed, e.g., by [RS94], whatever the setting is typed or not.
- The rule $(\mathcal{C}_{idem})$ $\mathcal{C}\ \lambda k.\mathcal{C}\, M \rightarrow \mathcal{C}\ \lambda k.(M)\, \lambda x.\mathcal{C}\ \lambda y.(k)\, x$ from Felleisen *et al* is disregarded, though needed to get a meaningful operational semantics: without $(\mathcal{C}_{idem})$, the calculus does not satisfy the property of *progress* that any closed term M which is not a value is reducible when evaluated in an abortive context [5].
- The motivation for keeping the rule $\mathcal{C}_{top}$ is difficult to understand. As it is presented, it adds no expressive power since it simply amounts to a sequence of two η-expansions. The original rule in Felleisen *et al* is $\mathcal{C}\, M \rightarrow \mathcal{C}\ \lambda k.(M)\, \lambda f.(\mathcal{A})(k)\ f$ and the justification of de Groote for removing the *abort* is that it is not needed for typing. This argument is sound from the operational point of view for $\mathcal{C}_L$ but not for $\mathcal{C}_{top}$: with *abort* disregarded from $\mathcal{C}_{top}$, we do not get the expected equivalence say, between $\mathcal{C}\ \lambda k.(f)(k)x$ and $\mathcal{C}\ \lambda k.(k)x$ for f of type $\perp \rightarrow \perp$.
- As noticed later on by Hofmann and Streicher [HS02], call-by-name $\lambda_{\mathcal{C}}$, such as defined in de Groote, is not expressive enough to support an equivalent to the renaming rule. The problem is that there is no local criterion for the needed distinction between continuation variables (that have to be captured by $\mathcal{C}$ so that a $\mathcal{C}_{idem}$ gets applicable) and ordinary variables (that have not to be captured by $\mathcal{C}$ because of the call-by-name discipline). The sole hope of an operationally complete semantics for call-by-name $\lambda_{\mathcal{C}}$ and of the ability to produce values out of closed computations is to reason at the level of the evaluation semantics, as done e.g. in Murthy [Mur91].

[5] This property is sometimes stated in λ-calculus under the form of a *unique context lemma* (as e.g. in [FF86]) telling that any term M which is not weak-head normal has the form $E[R]$ with R a redex, a redex which is the *head* redex of M.

Regarding $\lambda\mu$-calculus, de Groote's approach is also rather specific. His contribution is to provide a nice solution to the problem of having $\neg\neg A \to A$ proved by a closed term: in his variant of $\lambda\mu$-calculus, named terms and unnamed terms were not syntactically distinguished. De Groote's syntax for $\lambda\mu$-calculus was (in the following, we shall denote the set of terms of this syntax as $\Sigma_{\Lambda\mu}$):

$$\Sigma_{\Lambda\mu} \quad \ni \quad M, N ::= x \mid \lambda x.M \mid (M)\, M \mid \mu\alpha.M \mid [\alpha]M$$

De Groote also changed the typing system so that terms of the form $[\alpha]M$ have type $\bot$ and terms of the form $\mu\alpha.M$ expect M to be of type $\bot$. With this trick, there is no need anymore for a free variable to coerce M of type $\bot$ to a term of type, say A: it is enough to write $\mu\alpha.M$ where Parigot required to have $\mu\alpha.[\beta]M$ for β a free variable of type $\bot$ [6].

De Groote's modified syntax turned to be historically important and fruitful. Especially, this was the syntax used by the second author to prove a separability result [Sau05] which was wrong for the original syntax (as shown by David & Py [DP01]). At this time, the new syntax was not properly compared to the original calculus, and the comparison is actually not so trivial: this is the main purpose of the latter part of this paper.

$\lambda\mu\epsilon$-calculus. The status of the extended calculus introduced by de Groote is actually not absolutely clear since de Groote himself considers different variants of the extended calculus in his articles without analysing their relations.

In particular, in 1998, de Groote [dG98] defined an abstract machine for $\lambda\mu$-calculus in the extended syntax but considered new reduction rules. This alternative calculus, that we shall refer to as $\lambda\mu\epsilon$-calculus, is defined as follows: $\lambda\mu\epsilon$-terms are the elements of $\Sigma_{\Lambda\mu}$. $\lambda\mu\epsilon$-reduction is as follows:

Definition 2 ($\lambda\mu\epsilon$-calculus reduction) *$\lambda\mu\epsilon$-calculus reduction, which is written $\longrightarrow_{\lambda\mu\epsilon}$, is given by the following five reduction rules[7] presented in figure 4 where $|M|_\beta$ is the result of removing all free occurrences of β in M.*

The main novelty in $\lambda\mu\epsilon$-calculus – which extends Parigot's calculus – is rule (ϵ):

$$\mu\alpha.\mu\beta.M \to_\epsilon \mu\alpha.\,|M|_\beta$$

[6] $\neg\neg A \to A$ is inhabited by $\aleph' = \lambda y.\mu\alpha.(y)\lambda x.[\alpha]x$ for instance.

[7] De Groote attributes to Parigot the naming of rule μ. However, we could not trace this notation back to Parigot's writings.

$$
\begin{array}{lll}
(\beta)\ (\lambda x.M)N & \longrightarrow M\{N/x\} & \\
(\mu)\ (\mu\alpha.M)N & \longrightarrow \mu\beta.M\{[\beta](P)N/[\alpha]P\} & \text{if } \beta \notin FV(MN) \\
(\rho)\ \mu\gamma.[\beta]\mu\alpha.M & \longrightarrow \mu\gamma.M\{\beta/\alpha\} & \\
(\theta)\ \mu\alpha.[\alpha]M & \longrightarrow M & \text{if } \alpha \notin FV(M) \\
(\epsilon)\ \mu\alpha.\mu\beta.M & \longrightarrow \mu\alpha.|M|_\beta &
\end{array}
$$

Figure 4. $\lambda\mu\epsilon$-calculus reduction rules.

De Groote's motivation in adding this rule was to allow erasure of continuation variables of type $\bot$ [8] but without requiring the term to carry type information. He thus had to rely on syntactical constraints which force the appropriate type [9]: indeed, for $\mu\alpha.\mu\beta.M$ to be well typed, $\mu\beta.M$ is forced to be of type $\bot$ and thus β is also of type $\bot$.

Comment 2 *An early version of this rule was already discussed in the conclusion of his LPAR paper [dG94] comparing $\lambda\mu$-calculus and $\lambda_{\mathcal{C}}$-calculus, ϵ being viewed as useful for encoding the abort operator in $\lambda\mu$-calculus (see the discussion above).*

Moreover, rules closely related with ϵ are present in Ong and Selinger's presentations of $\lambda\mu$-calculus as we shall see below.

The introduction of rule ϵ requires several modifications on the other rules of the calculus. A constraint on reduction S_1 shall be added in order not to lose confluence, which leads de Groote to reformulate reduction S_1 as ρ-reduction [10] shown below:

$$\mu\gamma.[\beta]\mu\alpha.M \longrightarrow_\rho \mu\gamma.M\{\beta/\alpha\}.$$

In the remaining of this section, we shall *(i)* focus on other contributions on $\lambda\mu$ which are based on the extended syntax $\Sigma_{\Lambda\mu}$, and more particularly Ong's and Selinger's contributions, *(ii)* discuss the duality of computation approach

[8] He refers to this rule as "elimination of absurd weakening".

[9] According to his own words: "we have to define these notions of reduction at the level of untyped terms since we do not want the notion of type to play any dynamic part when evaluating a $\lambda\mu$-term".

[10] This constraint restricts the application of reduction ρ to redexes where Parigot's syntactical constraint $\mu[]$ is satisfied. Otherwise there is a critical pair involving ϵ and ρ which cannot converge as exemplified by $[\zeta][\delta]\mu\gamma.\mu\beta.\mu\alpha.M$:

$$
\begin{array}{l}
[\zeta][\delta]\underline{\mu\gamma.\mu\beta}.\mu\alpha.M \longrightarrow_\epsilon [\zeta][\delta]\mu\gamma.\mu\alpha.|M|_\beta \longrightarrow^2_\rho |M|_\beta\{\delta/\gamma\}\{\zeta/\alpha\} \\
[\zeta][\delta]\mu\gamma.\underline{\mu\beta.\mu\alpha}.M \longrightarrow_\epsilon [\zeta][\delta]\mu\gamma.\mu\beta.|M|_\alpha \longrightarrow^2_\rho |M|_\alpha\{\delta/\gamma\}\{\zeta/\beta\}.
\end{array}
$$

$$
\begin{array}{lll}
(\beta) & (\lambda x^A.M)\,N = M\,\{N/x\} & \\
(\eta) & \lambda x^A.(M)x = M & \text{if } x \notin FV(M) \\
(\mu{-}\beta) & [\beta^B]\mu\alpha^B.c = c\left\{\beta^B/\alpha^B\right\} & \\
(\mu{-}\eta) & \mu\alpha^B.[\alpha^B]M = M & \text{if } \alpha^B \notin FV(M) \\
(\zeta) & \mu\alpha^{A\to B}.c = \lambda x^A.\mu\beta^B.c\left\{[\beta^B](P)x/[\alpha^{A\to B}]P\right\} & \text{if } x^A, \beta^B \notin FV(c) \\
(\zeta^\perp) & \mu\alpha^{A\to\perp}.c = \lambda x^A.c\left\{(P)x/[\alpha^{A\to\perp}]P\right\} & \text{if } x^A \notin FV(c)
\end{array}
$$

Figure 5. Complete equational theory for Ong's $\lambda\mu$-calculus in modified syntax.

---

which made clear the duality between terms and evaluation contexts and of their respective substitutions which also led to considering a continuation constant tp of type $\perp$, and finally *(iii)* evidence the distinction between $\lambda\mu$ and $\Lambda\mu$ (that is, between the original and the extended syntax) in the untyped case by discussing the separation property in $\lambda\mu$.

2.3 From Ong (1996) to Selinger (2000): a study of the theory of typed $\lambda\mu$-calculus in modified syntax through semantical criteria

Ong studied call-by-name $\lambda\mu$-calculus from a categorical point of view. For this purpose, he was led to consider an equational theory for $\lambda\mu$-calculus that is complete with respect to the equations of the categorical model (a categorical model that is characterised as a fibred cartesian closed category). In particular, the semantics approach forces to consider η-rules (universal properties) for the two connectives considered, namely $A \to B$ and $\perp$. Because of the presence of two connectives, η-rules need to know the type of the term to η-expand, what requires explicitly typed reduction rules. Ong's syntax is:

$$
\begin{array}{lll}
B & ::= X \mid A \to A & \text{(non-empty types)} \\
A & ::= B \mid \perp & \text{(types)} \\
M & ::= x \mid \lambda x^A.M \mid (M)\,M \mid \mu\alpha^B.M \mid [\alpha^B]M & \text{(terms)}
\end{array}
$$

Besides providing an equational theory complete with respect to some categorical semantics, Ong improved on the way Parigot named the rules of $\lambda\mu$. The complete equational theory with Ong's genuine rule names is given in figure 5.

The advantage of Ong's rules $(\zeta)/(\zeta^\perp)$ compared to the original structural

reduction of Parigot is unclear to us. Indeed, while structural reduction was purely operational, Ong's rules (ζ) and ($\zeta^{\perp}$) introduce a bit of η-expansion. It can indeed be decomposed as an η-expansion together with a structural reduction (if $x^A, \beta^B \notin FV(c)$):

$$\mu\alpha^{A\to B}.c = \lambda x^A(\mu\alpha^{A\to B}.c)x = \lambda x^A.\mu\beta^B.c\left\{[\beta^B](P)x/[\alpha^{A\to B}]P\right\}$$

As a consequence, it breaks the clean separation between a set of operational rules and a set of purely observational rules (see figure 13 for a presentation where operational and observational fragments are clearly separated). Moreover, Ong's system is not minimal since η can be derived from (ζ) and (μ−η).

The motivation for forbidding continuation variables of type $\perp$ is unclear too: at the end, Ong's typed $\lambda\mu$-calculus can be conservatively extended using the following more uniform syntax:

$$\begin{array}{lll} A, B ::= X \mid \perp \mid A \to A & & \text{(types)} \\ M \quad ::= x \mid \lambda x^A.M \mid (M)\, M \mid \mu\alpha^B.M \mid [\alpha^B]M & & \text{(terms)} \end{array}$$

and by replacing ($\zeta^{\perp}$) with

$$(\eta_{\perp}) \quad [\alpha^{\perp}]M = M.$$

Both these peculiarities are solved in Selinger [Sel01], who, in addition to his contribution to the understanding of the semantical duality between call-by-name and call-by-value, switched back to structural reduction and explicitly supported continuation variables of type $\perp$. Moreover, Selinger slightly improved on the rationale for naming rules and insisted on formulating the rules on type derivations, thus avoiding problems such that ensuring the correctness of the type annotation in (β) or the validity of (η) when used from right to left.

Selinger's presentation of typed $\lambda\mu$-calculus in modified syntax is given in figure 6 (we drop the constructions and rules for pairs, sums and units) [11].

There is a canonical mapping of Selinger's typed terms to Ong's typed terms which is compositional except for $\mu\alpha.M$ and $[\alpha]M$ for α of type $\perp$ in which case the translation is M. Actually, we have the following proposition (implicit in Selinger's work) stating that Selinger's axiomatic is equivalent to Ong's axiomatic:

Proposition 3 *There is an equational correspondence between Ong's simply-typed $\Lambda\mu$ and Selinger's simply-typed $\Lambda\mu$.*

[11] Moreover, [Sel01] says B instead of $A \to B$ and we assume it is a typo.

$$
\begin{array}{lll}
(\beta_{\to}) & (\lambda x^A.M)\,N & = M\,\{N/x\} : B \\
(\eta_{\to}) & \lambda x^A.(M)x & = M : A \to B \quad [12] \qquad \text{if } x \notin FV(M) \\
(\zeta_{\to}) & (\mu\alpha^{A\to B}.M)N & = \mu\beta^B.M\left\{[\beta^B](P)N/[\alpha^{A\to B}]P\right\} \text{ if } \beta^B \notin FV(M,N) \\
(\beta_{\mu}) & [\beta]\mu\alpha^A.M & = M\,\{\beta/\alpha\} : \bot \\
(\eta_{\mu}) & \mu\alpha^A.[\alpha]M & = M : A \qquad \text{if } \alpha \notin FV(M) \\
(\beta_{\bot}) & [\alpha^{\bot}]M & = M : \bot
\end{array}
$$

Figure 6. Selinger's presentation of typed $\lambda\mu$-calculus in modified syntax.

PROOF: We briefly sketch the proof of the proposition. (ζ) is directly derivable from $(\eta_{\to})$ and $(\zeta_{\to})$ while $(\zeta^{\bot})$ can be derived from $(\eta_{\to})$, $(\zeta_{\to})$, $(\beta_{\bot})$ and (η_{μ}). In the other way round, $(\beta_{\bot})$ becomes an identity and $(\zeta_{\to})$ derives from (ζ) or $(\zeta^{\bot})$ and (β). ■

Let us briefly come back to $\lambda\mu\epsilon$-calculus: de Groote's ϵ rule is related with the $(\zeta^{\bot})$ rule by Ong and the $(\beta_{\bot})$ rule by Selinger [13] . However, contrarily to Ong's and Selinger's calculi de Groote's $\lambda\mu\epsilon$-calculus has a type-free presentation due to the use of the implicit type constraint of μ encapsulated terms.

One owes to Ong, together with Stewart with whom he provided a typed call-by-value variant of the modified syntax [OS97], to have introduced a notation $M[\beta, K/\alpha]$ generic over evaluation context K that lets suggest that Parigot's structural reduction is indeed a notion of substitution of evaluation contexts.

Hofmann and Streicher [HS97] later provided an alternative, more syntactic characterisation in terms of continuation-passing-style (cps) transformation of the completeness of Ong's equational theory of $\lambda\mu$-calculus in modified syntax. As noticed by Streicher and Reus [SR98], Plotkin's call-by-name cps-translation extended to $\lambda\mu$-calculus [14] is not appropriate as it validates $(\to)$, $(\mu_{\to})$ and (η_{μ}) but not $(\eta_{\to})$. Hofmann and Streicher's completeness result indeed relies on Lafont-Reus-Streicher's call-by-name cps-semantics [LRS93].

[13] Actually, in the typed case, ϵ is validated by Selinger's equational theory:
$\mu\alpha^A.\mu\beta^{\bot}.M =_{\beta_{\bot}} \mu\alpha^A.\mu\beta^{\bot}.|M|_{\beta} =_{\beta_{\bot}} \mu\alpha^A.\mu\beta^{\bot}.[\beta]\,|M|_{\beta} =_{\eta_{\mu}} \mu\alpha^A.\,|M|_{\beta}$.

[14] The translation is as follows:

$$
\begin{array}{llll}
[\![x]\!]_K & = (x)K & [\![\lambda x.M]\!]_K = (K)\lambda x.[\![M]\!] & [\![[\beta]M]\!] = [\![M]\!]_{k_\beta} \\
[\![(M)N]\!]_K = [\![M]\!]_{\lambda f.(f)\,[\![N]\!]_K} & [\![\mu\alpha.c]\!]_K & = (\lambda k_\alpha.[\![c]\!])\,K & [\![M]\!] \quad = \lambda k.[\![M]\!]_k
\end{array}
$$

In 2002, Hofmann and Streicher [HS02] showed that Ong and Stewart's semantics of call-by-value $\lambda\mu$-calculus (in modified syntax) is equivalent to Hofmann and Sabry-Felleisen's call-by-value $\lambda_{\mathcal{C}}$ semantics [Hof95,SF93] and hence complete with respect to Plotkin or Fisher's call-by-value continuation-passing-style semantics.

2.4 Looking at λμ-calculus from the duality-of-computation point of view

In 2000, Curien and the first author [CH00] designed a variant of $\lambda\mu$-calculus based on the syntax of Gentzen's sequent calculus. In sequent calculus, connectives are introduced by left or right introduction rules that echo to other introduction rules on the other side of the sequent. In sequent calculus, the computation is expressed in a purely structural way as a *cut* between a formula on the left and a formula on the right. Altogether, this contributes to establish a duality between the right-hand side and the left-hand side of a sequent.

Eliminating cuts necessarily breaks the symmetry of the calculus because one side has to get the priority over the other and the two dual ways to break the symmetry have been shown to correspond to the call-by-name/call-by-value duality, giving at the same time a purely syntactic emphasis of this duality.

In this computational interpretation of sequent calculus, right introduction rules correspond to constructors of terms and left introduction rules to constructors of evaluation contexts, leading to a duality between these notions and more generally to a reformulation of the concept of computation as an interaction happening at the frontier between a term and its evaluation context [15].

---

[15] For the record, the syntax of $\overline{\lambda}\mu\tilde{\mu}$-calculus is

$c ::= \langle t \| e \rangle$ (commands)
$t ::= x \mid \lambda x.t \mid \mu\alpha.c$ (terms)
$e ::= \alpha \mid t \cdot e \mid \tilde{\mu}x.c$ (evaluation contexts)

where $\tilde{\mu}x.c$ is a notation for the context $\mathbf{let}\ x = [\,]\ \mathbf{in}\ c$, $t \cdot e$ for the context $e[[\,]\,t]$ and α alone for the context $[\alpha]([\,])$. The reduction rules are

$(\mu)\quad \langle \mu\alpha.c \| e \rangle \to c\{e/\alpha\}$
$(\tilde{\mu})\quad \langle t \| \tilde{\mu}x.c \rangle \to c\{v/x\}$
$(\to)\quad \langle \lambda x.t \| t' \cdot e \rangle \to \langle t' \| \tilde{\mu}x.\langle t \| e \rangle \rangle$

with the call-by-name calculus obtained by giving priority to $\tilde{\mu}x.c$ in the critical pair $\langle \mu\alpha.c \| \tilde{\mu}x.c' \rangle$ and the call-by-value calculus obtained by giving instead priority to $\mu\alpha.c$. In practise, the construction $\tilde{\mu}x.c$ can be made implicit in the call-by-name fragment providing a correspondence with call-by-name $\lambda\mu$-calculus by informally identifying $M\,N_1\,\ldots\,N_n$ in the latter by $\mu\alpha.\langle M \| N_1 \cdot \ldots \cdot N_n \cdot \alpha \rangle$, for α fresh.

One of the consequences of the syntactic duality between terms and evaluation contexts and between call-by-name and call-by-value is that μ is revealed as the exact dual of an evaluation context **let** $x = [\,]$ **in** M that binds the program to which it is applied to x before continuing the computation with program M (an evaluation context which is commonly used in the context of call-by-value semantics). This allows to see that μ is a binder of evaluation contexts in the same way as a **let** $x = [\,]$ **in** M expression is a binder of terms. Especially, it appears that renaming and structural substitution are two complementary pieces of the same notion, namely the substitution of evaluation contexts.

As explored in [Her05], another consequence of the sequent calculus approach of $\lambda\mu$-calculus is the ability to make a clear distinction between the logical (= computational) rules and the observational rules.

Finally, by emphasising a syntactic duality between terms and evaluation contexts, [CH00] promotes the use of an explicit continuation constant for interpreting the elimination rule of $\bot$, as a parallel to the use of an explicit term constant for interpreting the introduction rule of $\top$.

2.5 Classical Logic vs. Minimal Classical Logic (2003)

In section 2.1, we saw that in Parigot's $\lambda\mu$-calculus it is not possible to build a closed proof-term for $\neg\neg A \to A$, which is not satisfactory from the proof-theoretic point of view. There is an alternative solution to the extended syntax in order to solve this problem: Ariola and Herbelin [AH03] (extended version with Sabry [AHS07a]) replace the "trick" of hiding continuation variables of type $\bot$ by the introduction in the syntax of a continuation constant tp of type $\bot$ (i.e. that expects an argument of type $\bot$). In particular, this clarifies that pure $\lambda\mu$-calculus is in correspondence with minimal classical logic (Peirce's law $((A \to B) \to A) \to A$ holds but $\bot \to A$ does not) and not with full classical logic (both Peirce's law and $\bot \to A$ hold, what amounts to have $\neg\neg A \to A$).

Comment 3 *Reconsidering [AH03], we think that Ariola and Herbelin approach can be improved by simply considering that $\bot$ is a connective and that the construction $[\mathsf{tp}]M$ is its elimination rule. This view also is obvious from the point of view of the duality of computation for which tp is the only constructor, in the class of evaluation contexts, of connective $\bot$.*

2.6 Separation property in $\lambda\mu$-calculus (2005)

In his seminal paper [Par92], Parigot remarks that "in $\lambda\mu$-calculus (contrary to λ-calculus), there are terms which give always the same result, indepen-

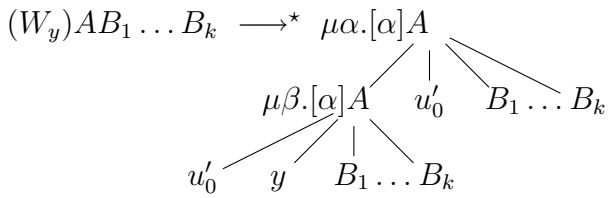

Figure 7. Counter-example to separation in $\lambda\mu$-calculus.

---

dently of the number of arguments they are applied to. For instance, the term $\tau = \lambda x.\lambda y.\mu\delta.[\phi](x)y$ is such that, for each $n \in \mathbb{N}, (\tau)xyz_1 \dots z_n \rightarrow^\star \mu\delta[\phi](x)y$." The situation is actually worse: separation (*aka.* Böhm theorem) fails in $\lambda\mu$-calculus, *i.e.* there exist non-equivalent terms that give the same results whatever context they are placed in.

Failure of separation was proved by David and Py [DP01] [16]. For this, they introduce new reduction rules to consider a confluent $\lambda\mu$-calculus with η-rules:

$$\begin{array}{lcll} \lambda x.(M)x & \longrightarrow & M & \text{if } x \notin FV(M) \\ \mu\alpha.c & \longrightarrow & \lambda x.\mu\beta.c\{[\beta](P)x/[\alpha]P\} & \text{if } x, \beta \notin FV(c) \end{array}$$

They then prove that separation fails in $\lambda\mu$ by finding a counter-example to separation. They exhibit two values $W_0, W_1 \in \Sigma_{\lambda\mu}$ [17] that are not equivalent for the equivalence relation induced by $\lambda\mu$-reduction rules (with extensionality) but that no context can distinguish.

The failure of separation in $\lambda\mu$-calculus may be understood as the fact that some separating contexts are missing in $\lambda\mu$ for separation to hold resulting in the impossibility of observing y in the term of footnote 17 as shown in figure 7. This led the second author to define $\Lambda\mu$-calculus [Sau05], an extension to $\lambda\mu$ for which Böhm theorem was proved. In $\Lambda\mu$-calculus, the validity of separation may be understood as the fact that the new contexts made available by the new syntax are sufficient to realise a Böhm Out.

$\Lambda\mu$-calculus syntax actually coincides with the extended syntax of de Groote and, up to the extensionality rules, the two calculi are identical:

Definition 4 ($\Lambda\mu$-calculus reduction) *$\Lambda\mu$-calculus reduction, which is written $\longrightarrow_{\Lambda\mu}$, is induced by the five reduction rules presented in figure 8* [18].

A separation result is stated with respect to a set of observables (in λ-calculus,

[16] The result was already in Py's thesis [Py98].
[17] The terms are obtained by substituting y by $0 = \lambda x, x'.x'$ and $1 = \lambda x, x'.x$

(β_T) $(\lambda x.M)N$	$\longrightarrow$	$M\{N/x\}$	
(η_T) $\lambda x.(M)x$	$\longrightarrow$	M	if $x \notin FV(M)$
(β_S) $[\beta]\mu\alpha.M$	$\longrightarrow$	$M\{\beta/\alpha\}$	
(η_S) $\mu\alpha.[\alpha]M$	$\longrightarrow$	M	if $\alpha \notin FV(M)$
(fst) $\mu\alpha.M$	$\longrightarrow$	$\lambda x.\mu\beta.M\{[\beta](P)x/[\alpha]P\}$	if $x, \beta \notin FV(M)$

Figure 8. $\Lambda\mu$-calculus reduction rules.

they are the $\beta\eta$-normal forms). Since *fst* is an expansion rule, there are very few normal forms in $\Lambda\mu$. We thus consider a set of $\Lambda\mu$-canonical normal forms [Sau05,Sau08b] as basic observables for separation:

Definition 5 *$M \in \Sigma_{\Lambda\mu}$ is in $\Lambda\mu$-canonical normal form if it is normal for $\beta_T\eta_T\beta_S\eta_S$ and it contains no subterm of the form $[\alpha]\lambda x.M$ nor $(\mu\alpha.M)N$.*

We can now state the separation result for $\Lambda\mu$-calculus [Sau05]:

Theorem 6 (Böhm theorem for $\Lambda\mu$) *Let $M, M' \in \Sigma_{\Lambda\mu}$. If $M \neq_{\Lambda\mu} M'$ and they are both $\Lambda\mu$-canonical normal forms, there exists a context $C[]$ such that:*

$$C[M] \longrightarrow^{\star}_{\Lambda\mu} \lambda x.\lambda y.x \qquad \textit{and} \qquad C[M'] \longrightarrow^{\star}_{\Lambda\mu} \lambda x.\lambda y.y.$$

By showing that the untyped modified syntax of $\lambda\mu$-calculus satisfies the Böhm separation theorem [Sau05] while the untyped original syntax did not [DP01], Saurin emphasised that, given a same set of reduction rules, the modified syntax was more expressive than the original syntax. This justified to give it the distinct name of $\Lambda\mu$-calculus, which is the name we keep for denoting the modified syntax in the rest of this article.

2.7 A global view of $\lambda\mu$-calculi.

In figure 9, we collect the various notations for reduction rules used in standard references that we discussed above, both in the standard and the modified syntax. The right-most column shows notations that shall be followed in the remaining sections of the present paper.

---

respectively in $W_y = \lambda x.\mu\alpha.[\alpha]((x)\mu\beta.[\alpha](x)U_0y)U_0$ with $U_0 = \mu\delta.[\alpha]\lambda z_1, z_2.z_2$.

[18] Notice that μ is not part of $\Lambda\mu$-calculus reduction system (but could be added): it can indeed be simulated by a *fst*-reduction followed by a β_T-reduction: $(\mu\alpha.M)N \longrightarrow_{fst} (\lambda x.\mu\alpha.M\{[\alpha](P)x/[\alpha]P\})N \longrightarrow_{\beta_T} \mu\alpha.M\{[\alpha](P)N/[\alpha]P\}$.

[Par91, Par93]	[Ong96]	[dG98]	[DP01]	[Sel01]	[Sau05]	[this paper]
(R_1)	(β)	(β)	(β)	$(\beta_{\rightarrow})$	(β_T)	$(\rightarrow)$
	(η)		(η)	$(\eta_{\rightarrow})$	(η_T)	$(\eta_{\rightarrow})$
(R_2)		(μ)	(μ)	$(\zeta_{\rightarrow})$		$(\mu_{\rightarrow})$
(S_1)	$(\mu - \beta)$	(ρ)	(ρ)	(β_{μ})	(β_S)	(μ_{var})
(S_2)	$(\mu - \eta)$	(θ)	(θ)	(η_{μ})	(η_S)	(η_{μ})
(S_3)	(ζ)		(ν)		(fst)	
	$(\zeta^{\perp})$	(ϵ)		$(\beta_{\perp})$		$(id_{\perp})$

Figure 9. Notations for $\lambda\mu$-reductions in the literature.

3 Connecting the original and the modified syntax of $\lambda\mu$-calculus in the typed case

Connecting the original syntax based on two syntactical categories of terms and the approaches based on the modified syntax with a single category of term requires in the typed case two extensions of the original syntax. Following David and Py, we first complete Parigot's reduction system with η-rules while, in a second step, we add to $\lambda\mu$-calculus a $\perp$ connective.

As a first step, we remind some results on the notion of cps-completeness that we use for characterising the minimum of equations to be valid in a sensible axiomatic theory of $\lambda\mu$-calculus.

3.1 Cps-complete extension of Parigot's $\lambda\mu$-calculus

Ong [Ong96] studied $\lambda\mu$-calculus as an equational theory with η-rules for implication but it was for the modified syntax *and* in a typed setting. Hofmann and Streicher [HS97] showed that this same equational theory was also sound and complete with respect to Lafont-Reus-Streicher's call-by-name continuation-passing-style semantics [LRS93].

For the original syntax, the extension with η for implication was considered by David and Py [DP01] who showed that Böhm separation theorem does not hold. Later on, Fujita adapted Hofmann and Streicher's soundness and completeness proof from the modified syntax to the original syntax and in an untyped setting. The corresponding equational theory is presented in figure 10.

$$
\begin{array}{lll}
(\rightarrow) & (\lambda x.M)N = M\{N/x\} & \\
(\eta_{\rightarrow}) & \lambda x.(M)x = M & \text{if } x \text{ not free in } M \\
(\mu_{\rightarrow}) & (\mu\alpha.c)N = \mu\beta.c\{[\beta](P)\,N/[\alpha]P\} & \text{if } \beta \text{ not free in } c, N \\
(\mu_{var}) & [\beta]\mu\alpha.c = c\{\beta/\alpha\} & \\
(\eta_{\mu}) & \mu\alpha.[\alpha]M = M & \text{if } \alpha \text{ not free in } M
\end{array}
$$

Figure 10. CPS-Complete equational theory for Parigot's $\lambda\mu$-calculus.

Proposition 7 ([Fuj03]) *The equational theory is validated by its call-by-name continuation-passing-style (cps) to λ-calculus with pairs.*

3.2 Extension of the original syntax with a constructor for the $\bot$ connective

So as to clarify the connection with the modified syntax of $\lambda\mu$-calculus, we show the extension of Parigot's $\lambda\mu$-calculus to the $\bot$ connective and its inference rules. Following [AH03], this extension is obtained by adding a construction $[\mathsf{tp}]M$ where tp is a continuation constant typable of type $\bot$.

$$
\begin{array}{rcl}
\Sigma_{\lambda\mu\mathsf{tp}} & \ni & M ::= x \mid \lambda x.M \mid (M)\,M \mid \mu\alpha.c \\
& & c ::= [\alpha]M \mid [\mathsf{tp}]M
\end{array}
$$

In the typed case, extensionality rules are $[\mathsf{tp}]M =_{\eta_{\bot}} [\alpha]M$ for the $\bot$ connective[19] and $\lambda x.(M)x =_{\eta_{\rightarrow}} M$ (for x not free in M) for the $\rightarrow$ connective. While $\eta_{\rightarrow}$ can be easily considered in the untyped case, it is not the case of $\eta_{\bot}$ which is hardly manageable without type information. As a consequence, we shall stay in the typed case at the moment, as given on the left column of Figure 13 where a rule $M = N : A$ denotes a transformation between derivations of the form $\Gamma \vdash M : A; \Delta$ and $\Gamma \vdash N : A; \Delta$ for some Δ and Γ.

3.3 Cps-complete presentation of the typed modified syntax of $\lambda\mu$-calculus ($\Lambda\mu^{\rightarrow}_{st}$)

Let us use the name $\Lambda\mu_{st}$ for Selinger's axiomatics of the simply-typed modified syntax of $\lambda\mu$-calculus [Sel01]. This calculus is recalled on the right column of Figure 13. The η-reduction rules related to connectives are applicable when

[19] This rule expresses that the only "continuation" of type $\bot$ is tp, in the same way as extensionality for $\top$ is $x = () : \top$, where $()$ is the canonical proof inhabiting $\top$.

$\lambda\mu^{\perp}_{st}$		$\Lambda\mu^{\perp}_{st}$
$\Gamma \vdash c; \Delta$	$\longleftrightarrow$	$\Gamma \vdash c : \perp; \Delta$
$\Gamma \vdash M : A; \Delta$	$\longleftrightarrow$	$\Gamma \vdash M : A; \Delta$
$[\alpha]M$	$\longmapsto$	$[\alpha]M$
$[\mathsf{tp}]M$	$\longmapsto$	$[\mathsf{tp}]M$
$\mu\alpha.c$	$\longmapsto$	$\mu\alpha.c$
$\mu\alpha.[\mathsf{tp}]M$	$\longmapsfrom$	$\mu\alpha.M$
$\mu\_.[\alpha]M$	$\longmapsfrom$	$[\alpha]M$
$\mu\_.[\mathsf{tp}]M$	$\longmapsfrom$	$[\mathsf{tp}]M$

Figure 11. Translations between $\lambda\mu^{\perp}_{st}$ and $\Lambda\mu^{\perp}_{st}$.

$\Lambda\mu^{\perp}_{st}$		$\Lambda\mu_{st}$
$\Gamma \vdash M : A; \Delta$	$\longleftrightarrow$	$\Gamma \vdash M : A; \Delta$
$[\alpha]M$	$\longleftrightarrow$	$[\alpha]M$
$\mu\alpha.M$	$\longleftrightarrow$	$\mu\alpha.M$
$[\mathsf{tp}]M$	$\longmapsto$	M

Figure 12. Translations between $\Lambda\mu^{\perp}_{st}$ and $\Lambda\mu_{st}$.

both terms are typable of the same type. Moreover, to move from one calculus to another, the following translations shall be applied:

Definition 8 *Translations between $\lambda\mu^{\perp}_{st}$, $\Lambda\mu^{\perp}_{st}$ and $\Lambda\mu_{st}$ are defined as follows:*

- *translations between $\lambda\mu^{\perp}_{st}$ and $\Lambda\mu^{\perp}_{st}$ are presented in figure 11. They are written $|\ |_{\Lambda\mu^{\perp}}$ for the translation from $\lambda\mu^{\perp}_{st}$ to $\Lambda\mu^{\perp}_{st}$ and $|\ |_{\lambda\mu^{\perp}}$ for the reverse translation.*
- *translations between $\Lambda\mu^{\perp}_{st}$ and $\Lambda\mu_{st}$ are presented in figure 12.*

3.4 Modified syntax extended with a constructor for the $\perp$ connective

As in subsection 3.2, we extend $\Lambda\mu_{st}$ with an explicit continuation constant tp witnessing the $\perp$ elimination rule. This will ease the comparison with $\lambda\mu^{\perp}_{st}$. The resulting calculus, which is shown on the central column of Figure 13, is equivalent to Selinger's calculus. More precisely, we have:

<table>
<tr><th>$\lambda\mu^{\perp}_{st}$</th><th>$\Lambda\mu^{\perp}_{st}$</th><th>$\Lambda\mu_{st}$</th></tr>
<tr><td colspan="3">Syntax</td></tr>
<tr><td colspan="2">$\varsigma ::= \alpha \mid \mathsf{tp}$</td><td>$\varsigma ::= \alpha$</td></tr>
<tr><td>$M, N ::= x \mid \lambda x.M \mid (M)\,M \mid \mu\alpha.c \qquad c ::= [\varsigma]M$</td><td colspan="2">$M, N, c ::= x \mid \lambda x.M \mid (M)\,M \mid \mu\alpha.M \mid [\varsigma]M$</td></tr>
<tr><td colspan="3">Operational rules</td></tr>
<tr><td colspan="3">$(\rightarrow)\ (\lambda x.M)\,N = M\{N/x\} \qquad (\mu_{\rightarrow})\ (\mu\alpha.c)\,N = \mu\beta.\{[\beta](P)N/[\alpha]P\} \qquad (\mu_{var})\ [\varsigma]\mu\alpha.c = c\{\varsigma/\alpha\}$</td></tr>
<tr><td colspan="3">Observational rules</td></tr>
<tr><td colspan="3">$(\eta_{\mu})\ \mu\alpha.[\alpha]x = x : A \qquad (\eta_{\rightarrow})\ \lambda x.(y)\,x = y : A \rightarrow B$</td></tr>
<tr><td>$(\eta_{\perp})\ [\mathsf{tp}]M = [\alpha]M$</td><td>$(\eta_{\perp})\ [\mathsf{tp}]M = [\alpha]M : \perp$
$(id_{\perp})\ [\varsigma]M = M : \perp$</td><td>$(id_{\perp})\ [\varsigma]M = M : \perp$</td></tr>
<tr><td colspan="3">Typing rules</td></tr>
<tr><td colspan="3">$(A ::= X \mid A \rightarrow A \mid \perp)$</td></tr>
<tr><td colspan="3">$\dfrac{\Gamma \vdash M : A \rightarrow B; \Delta \qquad \Gamma \vdash N : A; \Delta}{\Gamma \vdash (M)N : B; \Delta} \qquad \dfrac{}{\Gamma, x : A \vdash x : A; \Delta} \qquad \dfrac{\Gamma, x : A \vdash M : B; \Delta}{\Gamma \vdash \lambda x.M : A \rightarrow B; \Delta}$</td></tr>
<tr><td>$\dfrac{\Gamma \vdash M : A; \Delta, \alpha : A}{\Gamma \vdash [\alpha]M; \Delta, \alpha : A} \qquad \dfrac{\Gamma \vdash c; \Delta, \alpha : A}{\Gamma \vdash \mu\alpha.c : A; \Delta}$
$\dfrac{\Gamma \vdash M : \perp; \Delta}{\Gamma \vdash [\mathsf{tp}]M; \Delta}$</td><td>$\dfrac{\Gamma \vdash M : A; \Delta, \alpha : A}{\Gamma \vdash [\alpha]M : \perp; \Delta, \alpha : A} \qquad \dfrac{\Gamma \vdash M : \perp; \Delta}{\Gamma \vdash [\mathsf{tp}]M : \perp; \Delta}$</td><td>$\dfrac{\Gamma \vdash M : \perp; \Delta, \alpha : A}{\Gamma \vdash \mu\alpha.M : A; \Delta}$</td></tr>
</table>

In order to minimise the redundancy in the table and to show as much of the common elements between the calculi as possible, we adopt the following writing conventions: when some parts of a definition are common to several calculi, we have merged associated columns by removing the vertical bar between two columns. For instance, in defining the syntax of the calculi, it shall be read that both $\lambda\mu^{\perp}_{st}$ and $\Lambda\mu^{\perp}_{st}$ have a syntactical construct ς which is either a continuation variable, α, or a toplevel constant, tp, while $\Lambda\mu_{st}$ allows ς to denote only continuation variables. On the other hand, the definition of terms involve two levels in $\lambda\mu^{\perp}_{st}$ (following Parigot's constraint) while it is not the case for $\Lambda\mu^{\perp}_{st}$ and $\Lambda\mu_{st}$ (following extended syntax). Moreover the operational rules are common two the three calculi: the three columns are merged.

Figure 13. Defining and connecting $\lambda\mu^{\perp}_{st}$, $\Lambda\mu^{\perp}_{st}$ and $\Lambda\mu_{st}$: syntax, operational rules, observational rules and typing rules.

Proposition 9 *$\Lambda\mu_{st}$ and $\Lambda\mu^{\perp}_{st}$ are in equational correspondence.*

PROOF: The sketch of the proof is as follows. First, notice that the η-rule for $\perp$ is a consequence of the property $(id_{\perp})$ that the only construction from $\perp$ to $\perp$ is the identity. Moreover, there is a retract from $\Lambda\mu^{\perp}_{st}$ to $\Lambda\mu_{st}$ that erases $[\mathsf{tp}]$. This retracts maps $(\eta_{\perp})$ steps to $(id_{\perp})$ steps. ■

We are now ready to relate $\Lambda\mu^{\perp}_{st}$ and $\lambda\mu^{\perp}_{st}$ and hence $\Lambda\mu_{st}$ and $\lambda\mu^{\perp}_{st}$. Again, there is a retract from $\Lambda\mu^{\perp}_{st}$ to $\lambda\mu^{\perp}_{st}$ that inserts a $\mu\_$ step (where "$\_$" denotes an arbitrary fresh evaluation context variable of type $\perp$) in front of any derivation of the form $[\varsigma]M$ not already prefixed by a $\mu\alpha$, and inserts a $[\mathsf{tp}]$ below any $\mu\alpha$ whose argument is not already prefixed by a $[\beta]$. We have:

Proposition 10 *$\lambda\mu^{\perp}_{st}$ and $\Lambda\mu^{\perp}_{st}$ are in equational correspondence.*

PROOF: The translation of $(id_{\perp})$ in $\lambda\mu^{\perp}_{st}$ is either $\mu\beta.[\varsigma]M = M : \perp$ (for β fresh) or $[\varsigma]M = [\mathsf{tp}]M : \perp$, depending on whether the equation is used in the immediate scope of a μ or not. The validity of $(id_{\perp})$ thus derives in the first case from $(\eta_{\perp})$ together with (η_{μ}) so as to get $\mu\beta.[\varsigma]M = \mu\beta.[\mathsf{tp}]M = \mu\beta.[\beta]M = M$ and derives directly from $(\eta_{\perp})$ in the second case. ■

Finally, the equational correspondence between $\lambda\mu^{\perp}_{st}$ and $\Lambda\mu_{st}$ is an immediate corollary of propositions 9 and 10:

Corollary 11 *$\lambda\mu^{\perp}_{st}$ and $\Lambda\mu_{st}$ are in equational correspondence.*

3.5 The operational correspondence between $\lambda\mu^{\perp}_{st}$ and $\Lambda\mu_{st}$

If we omit the observational rules, and in particular the ones related to $\perp$, then $\lambda\mu^{\perp}_{st}$ gets more distinct equivalence classes of terms than $\Lambda\mu^{\perp}_{st}$, but in a non essential way. If we write $\rightarrow$ for the left-to-right orientation of the operational rules, then the following holds:

Proposition 12

- *For M and N in $\lambda\mu^{\perp}_{st}$, $M \rightarrow N$ iff $|M|_{\Lambda\mu^{\perp}} \rightarrow |N|_{\Lambda\mu^{\perp}}$;*
- *For M and N in $\Lambda\mu^{\perp}_{st}$, if $M \rightarrow N$ then (μ_{var}) can be postponed and, if we exclude (μ_{var}) steps then $M \rightarrow N$ iff $|M|_{\lambda\mu^{\perp}} \rightarrow |N|_{\lambda\mu^{\perp}}$.*

PROOF: The first equivalence is not problematic, we concentrate on the second equivalence. In this case, (μ_{var}) shall be avoided. Indeed, considering (μ_{var}), the equivalence would fail for $|\mu\alpha.\mu\beta.x|_{\lambda\mu^{\perp}} = \mu\alpha.[\mathsf{tp}]\mu\beta.[\mathsf{tp}]x$ which reduces via (μ_{var}) to $\mu\alpha.[\mathsf{tp}]x = |\mu\alpha.x|_{\lambda\mu^{\perp}}$ and obviously $\mu\alpha.\mu\beta.x \not\rightarrow \mu\alpha.x$.

Postponement of (μ_{var}) holds because we are in the typed case: in the untyped case, this is not true for the extended syntax since a (μ_{var}) reduction may unlock a β-redex as shown by $([\beta]\mu\alpha.\lambda x.(x)x)\lambda x.(x)x \to (\lambda x.(x)x)\lambda x.(x)x$. ■

We can also show that $\lambda\mu^{\perp}$ and $\Lambda\mu$ are equivalent, even in the untyped case, if we change (μ_{var}) to

$$(\mu_{var}{}')\quad \mu\gamma[\beta]\mu\alpha.c = \mu\gamma.c\,\{\beta/\alpha\}$$

since then, all uses of $(\mu_{var}{}')$ in $\Lambda\mu^{\perp}_{st}$ are simulated in $\lambda\mu^{\perp}_{st}$.

The present section has been devoted to the investigation of the typed case: we established equational correspondence results between three calculi with an explicit representation of the toplevel as a toplevel continuation constant tp of type $\perp$. Actually, our results amount to show that in the typed setting, the original syntax and the modified syntax are essentially equivalent, as soon as one considers both an explicit construction for $\perp$ and observational rules.

We shall now turn our attention to the untyped setting.

4 Connecting the original and the modified syntax of $\lambda\mu$-calculus in the untyped case.

In this section, we first recall the connections between $\Lambda\mu$-calculus and call-by-name $\lambda\mu\widehat{\mathsf{tp}}$-calculus introduced in [HG08]. Then, we analyse $\lambda\mu\epsilon$ according to the same methodology: we build an equational correspondence with a variant of $\lambda\mu\widehat{\mathsf{tp}}$.

4.1 Equational correspondence between $\Lambda\mu$-calculus and call-by-name $\lambda\mu\widehat{tp}$

Introduced in [AHS04,AHS07b], the $\lambda\mu\widehat{\mathsf{tp}}$-calculus is a fine-grained calculus for delimited continuations of which a call-by-name version has been introduced by the first author and Ghilezan [HG08]. We recall the syntax of $\lambda\mu\widehat{\mathsf{tp}}$-calculus in the following definition:

Definition 13 ($\lambda\mu\widehat{\mathsf{tp}}$-terms) *$\lambda\mu\widehat{tp}$-terms ($M, N, \cdots \in \Sigma_{\lambda\mu\widehat{\mathsf{tp}}}$) are given by the following syntax:*

$$\begin{array}{rcl} \Sigma_{\lambda\mu\widehat{\mathsf{tp}}} & \ni & M, N ::= x \mid \lambda x.\, M \mid (M)M \mid \mu q.c \\ & & c ::= [q]M \qquad q ::= \alpha \mid \widehat{tp} \end{array}$$

$\Pi : \Sigma_{\Lambda\mu} \longmapsto \Sigma_{\lambda\mu\widehat{\mathsf{tp}}}$	$\Sigma : \Sigma_{\lambda\mu\widehat{\mathsf{tp}}} \mapsto \Sigma_{\Lambda\mu}$
$\Pi(x) \triangleq x$ $\Pi(\lambda x.M) \triangleq \lambda x.\Pi(M)$ $\Pi((M)N) \triangleq (\Pi(M))\Pi(N)$ $\Pi(\mu\alpha.M) \triangleq \mu\alpha.[\widehat{\mathsf{tp}}]\Pi(M)$ $\Pi([\alpha]M) \triangleq \mu\widehat{\mathsf{tp}}.[\alpha]\Pi(M)$	$\Sigma(x) \triangleq x$ $\Sigma(\lambda x.M) \triangleq \lambda x.\Sigma(M)$ $\Sigma((M)N) \triangleq (\Sigma(M))\Sigma(N)$ $\Sigma(\mu\alpha.[\beta]M) \triangleq \mu\alpha.[\beta]\Sigma(M)$ $\Sigma(\mu\alpha.[\widehat{\mathsf{tp}}]M) \triangleq \mu\alpha.\Sigma(M)$ $\Sigma(\mu\widehat{\mathsf{tp}}.[\alpha]M) \triangleq [\alpha]\Sigma(M)$ $\Sigma(\mu\widehat{\mathsf{tp}}.[\widehat{\mathsf{tp}}]M) \triangleq \Sigma(M)$

Figure 14. Translations between $\Lambda\mu$ and $\lambda\mu\widehat{\mathsf{tp}}$ [HG08].

While in the previous section we were considering a *constant* tp, things shall now be different in the rest of the paper. Indeed, in $\Sigma_{\lambda\mu\widehat{\mathsf{tp}}}$, $\widehat{\mathsf{tp}}$ is a toplevel continuation *variable* of a special kind. First, contrarily to the case of $\Sigma_{\lambda\mu\mathsf{tp}}$, one can abstract over $\widehat{\mathsf{tp}}$ using $\mu\widehat{\mathsf{tp}}$. while tp *could not be bound*. The second main difference is that $\widehat{\mathsf{tp}}$ will be considered as a *dynamic variable*: it will be dynamically bound by the closest encapsulating $\mu\widehat{\mathsf{tp}}$. As a conclusion, while substitution in the previous calculi involved α-conversion in order to avoid variable-capture, the substitution of $\widehat{\mathsf{tp}}$ for a usual continuation variable α will not imply such a conversion. As an example:

$$(\lambda x.\mu\widehat{\mathsf{tp}}.[\alpha].x)\left\{\widehat{\mathsf{tp}}/\alpha\right\} \triangleq \lambda x.\mu\widehat{\mathsf{tp}}.\left[\widehat{\mathsf{tp}}\right].x$$

This shall be crucial in building the correspondence with $\Lambda\mu$-calculus and $\lambda\mu\epsilon$-calculus. In particular, $\lambda\mu\mathsf{tp}$ is not equationally correspondent to $\Lambda\mu$-calculus in the untyped case for the following reason: $[\alpha][\beta]\mu\gamma.\mu\delta.M$ would translate into $\mu\zeta.[\alpha]\mu\zeta.[\beta]\mu\gamma.[\mathsf{tp}]\mu\delta.[\mathsf{tp}]M' =_{\mu_{var}} \mu\zeta.[\mathsf{tp}]M'\{\beta/\gamma\}\{\mathsf{tp}/\delta\}$. The only way to get this correspondence back would be to restrict μ_{var}/ρ as described in the previous section.

The equational theory of call-by-name $\lambda\mu\widehat{\mathsf{tp}}$-calculus is:

Definition 14 (CBN $\lambda\mu\widehat{\mathsf{tp}}$ equational theory) *$\lambda\mu\widehat{tp}$-equational theory is given by the equations presented in figure 15.*

$\Lambda\mu$-calculus and $\lambda\mu\widehat{\mathsf{tp}}$-calculus are in equational correspondence as proved in [HG08]:

Proposition 15 (Equational correspondence between $\Lambda\mu$ and $\lambda\mu\widehat{\mathsf{tp}}$) *Let $M, N \in \Sigma_{\Lambda\mu}$ and $M', N' \in \Sigma_{\lambda\mu\widehat{tp}}$. Then one has:*

- *If $M =_{\Lambda\mu} N$ then $\Pi(M) =_{\lambda\mu\widehat{tp}} \Pi(N)$;*
- *If $M' =_{\lambda\mu\widehat{tp}} N'$ then $\Sigma(M') =_{\Lambda\mu} \Sigma(N')$;*

$$
\begin{array}{llcll}
(\rightarrow) & (\lambda x.\, M)\ N & = & M\{N/x\} & \\
(\eta_{\rightarrow}) & \lambda x.\, (M)x & = & M & \text{if } x \text{ is not free in } M \\
(\mu_{\rightarrow}) & (\mu\alpha.\, c)\ M & = & \mu\beta.c\{[\beta](P)M/[\alpha]P\} & \text{if } \beta \text{ is not free in } c, M \\
(\mu^n_{var}) & [\beta]\mu\alpha.c & = & c\{\beta/\alpha\} & \\
(\eta_{\mu}) & \mu\alpha.[\alpha]M & = & M & \text{if } \alpha \text{ is not free in } M \\
(\mu_{\widehat{\mathsf{tp}}}) & \left[\widehat{\mathsf{tp}}\right]\mu\widehat{\mathsf{tp}}.c & = & c & \\
(\eta_{\widehat{\mathsf{tp}}}) & \mu\widehat{\mathsf{tp}}.\left[\widehat{\mathsf{tp}}\right]M & = & M & \text{even if } \widehat{\mathsf{tp}} \text{ occurs free in } M
\end{array}
$$

Figure 15. $\lambda\mu\widehat{\mathsf{tp}}$-calculus equational theory.

- $\Sigma(\Pi(M)) = M$;
- $\Pi(\Sigma(M')) =_{\mu_{\widehat{tp}}} M'$.

With Π *and* Σ*, the translations between* $\Lambda\mu$ *and* $\lambda\mu\widehat{tp}$ *that are given in figure 14.*

In the remaining of this section, we study a similar result for $\lambda\mu\epsilon$.

4.2 An alternative call-by-name delimited control calculus, $\lambda\mu\widehat{tp}\ell$

$\Lambda\mu$ (through its equational correspondence with $\lambda\mu\widehat{\mathsf{tp}}$) is not the only call-by-name calculus with delimited control. Ongoing work with Ghilezan [HGS09] is providing a uniform classification of calculi with delimited control which arise from choices among critical pairs (two CBN and two CBV calculi). In particular, $\lambda\mu\widehat{\mathsf{tp}}\ell$ is an alternative CBN calculus in which the toplevel continuation variable $\widehat{\mathsf{tp}}$ behaves as a regular linear evaluation context: in $\lambda\mu\widehat{\mathsf{tp}}\ell$, one of the changes is that rule μ^n_{var} is replaced by μ_{var}:

$$[q]\mu\alpha.c \longrightarrow c\{q/\alpha\}$$

allowing to substitute $\widehat{\mathsf{tp}}$ for a usual continuation variable.

The translation of $\mu\alpha.\mu\beta.M$ in $\Sigma_{\lambda\mu\widehat{\mathsf{tp}}}$, $\mu\alpha.\left[\widehat{\mathsf{tp}}\right]\mu\beta.\left[\widehat{\mathsf{tp}}\right]\Pi(M)$, contains a μ_{var} redex and thus reduces to $\mu\alpha.\left[\widehat{\mathsf{tp}}\right]\Pi(M)\left\{\widehat{\mathsf{tp}}/\beta\right\}$. Similarity with ϵ-reduction in $\lambda\mu\epsilon$-calculus is striking and this is what we shall investigate in the following subsections: can $\Pi(M)\left\{\widehat{\mathsf{tp}}/\beta\right\}$ be related to $\Pi(|M|_{\beta})$? We shall now introduce $\lambda\mu\widehat{\mathsf{tp}}\ell$ and see to what extent this may simulate $\lambda\mu\epsilon$. This shall finally lead us to present an alternative $\lambda\mu\widehat{\mathsf{tp}}$-calculus, called $\lambda\mu\widehat{\mathsf{tp}}\epsilon$-calculus, which is in

$$
\begin{array}{lll}
(\rightarrow) & (\lambda x.\, M)\; N \longrightarrow M\,\{N/x\} & \\
(\mu_{\rightarrow}) & (\mu\alpha.\, c)\; M \longrightarrow \mu\beta.c\,\{[\beta](P)M/[\alpha]P\} & \text{if } \beta \text{ is not free in } c, M \\
(\mu_{var}) & [q]\mu\alpha.c \longrightarrow c\,\{q/\alpha\} & \\
(\eta_{\mu}) & \mu\alpha.[\alpha]M \longrightarrow M & \text{if } \alpha \text{ is not free in } M \\
(\mu_{\widehat{\mathsf{tp}}}) & \left[\widehat{\mathsf{tp}}\right]\mu\widehat{\mathsf{tp}}.c \longrightarrow c & \\
(\eta^{v}_{\widehat{\mathsf{tp}}}) & \mu\widehat{\mathsf{tp}}.\left[\widehat{\mathsf{tp}}\right]V \longrightarrow V & \text{(even if } \widehat{\mathsf{tp}} \text{ occurs free in } V)
\end{array}
$$

Figure 16. $\lambda\mu\widehat{\mathsf{tp}}\ell$-calculus reduction rules.

equational correspondence with $\lambda\mu\epsilon$.

Terms of $\lambda\mu\widehat{\mathsf{tp}}\ell$ are in $\Sigma_{\lambda\mu\widehat{\mathsf{tp}}}$ and we additionally consider a set of values:

$$V ::= \lambda x.M \mid \mu\widehat{\mathsf{tp}}.c$$

Definition 16 ($\lambda\mu\widehat{\mathsf{tp}}\ell$-reduction) *$\lambda\mu\widehat{tp}\ell$-reduction system is defined by the six reductions presented in figure 16.*

4.3 Comparing $\lambda\mu\epsilon$ with $\lambda\mu\widehat{tp}\ell$

A quick analysis reveals that it is not always the case that if $M, N \in \Sigma_{\Lambda\mu}$ are such that $M \longrightarrow_{\epsilon} N$ then $\Pi(M) =_{\lambda\mu\widehat{\mathsf{tp}}\ell} \Pi(N)$. Indeed, in $\lambda\mu\epsilon$, an ϵ-reduction may unlock a hidden μ-redex as shown in the following examples:

$$M = \mu\alpha.\mu\beta.([\beta]\mu\gamma.N)P \longrightarrow_{\epsilon} \mu\alpha.(\mu\gamma.\,|N|_{\beta})\,|P|_{\beta}$$

whereas

$$\Pi(M) \longrightarrow_{\lambda\mu\widehat{\mathsf{tp}}\ell} \mu\alpha.\left[\widehat{\mathsf{tp}}\right](\mu\widehat{\mathsf{tp}}.\left[\widehat{\mathsf{tp}}\right]\mu\gamma.\left[\widehat{\mathsf{tp}}\right]\Pi(N)\left\{\widehat{\mathsf{tp}}/\beta\right\})\Pi(P)\left\{\widehat{\mathsf{tp}}/\beta\right\}$$

and the variable $\widehat{\mathsf{tp}}$ should be substituted to γ, thus forbidding to simulate the μ-reduction made available in $\lambda\mu\epsilon$ [20]. Thus, while in $\lambda\mu\epsilon$ an ϵ rule may unlock a μ-reduction, in $\lambda\mu\widehat{\mathsf{tp}}\ell$, the substitution of continuation variables by toplevel continuation variables may be propagated through continuation variables via the μ_{var} rule.

[20] The problem does not occur with a hidden β-redex (such as with term $M' = \mu\alpha.\mu\beta.([\beta]\lambda x.N)P$) since one obtains in $\lambda\mu\widehat{\mathsf{tp}}\ell$ the term $\mu\alpha.\left[\widehat{\mathsf{tp}}\right](\mu\widehat{\mathsf{tp}}.\left[\widehat{\mathsf{tp}}\right]\lambda x.\Pi(N)\left\{\widehat{\mathsf{tp}}/\beta\right\})\Pi(P)\left\{\widehat{\mathsf{tp}}/\beta\right\}$ to which a reduction $\eta^{v}_{\widehat{\mathsf{tp}}}$ can be applied because $\lambda x.\Pi(N)\left\{\widehat{\mathsf{tp}}/\beta\right\}$ is a value.

Since neither $\lambda\mu\epsilon$ nor $\lambda\mu\widehat{\mathsf{tp}}\ell$ can directly simulate the other calculus, one shall try to compare variants of the calculi. One option is to require a restriction on the ϵ-rule to the cases where it cannot unlock a hidden μ-redex. It is actually fairly simple to do: consider a term of the form $\mu\alpha.\mu\beta.M$ and consider all the subterms of M of the form $[\beta]N$. If N is $\lambda x.P$, there is no problem, if it is $[\gamma]P$, there is no problem either. On the contrary, if N is either of the form $\mu\alpha.P$, $(P)Q$ or x, then N is or may eventually reduce to a μ-abstracted term and the problem would occur. We thus consider the following restriction to ϵ-reduction:

$$\mu\alpha.\mu\beta.M \longrightarrow_{\epsilon_v} \mu\alpha.\,|M|_\beta \qquad \text{iff all free } \beta \text{ in } M \text{ occur at subterms } [\beta]V.$$

with $V ::= \lambda x.M \mid [\alpha]M$. In this case, we are sure that ϵ will not unlock a hidden μ-redex.

The resulting reduction is denoted as $\lambda\mu\epsilon_v$. One can thus have a simulation result:

Proposition 17 *For any $M, N \in \Sigma_{\Lambda\mu}$, if $M \longrightarrow_{\lambda\mu\epsilon_v} N$ then $\Pi(M) =_{\lambda\mu\widehat{\mathsf{tp}}\ell} \Pi(N)$.*

Proof: Indeed, every reduction in $\lambda\mu\epsilon_v$ can easily be simulated in $\lambda\mu\widehat{\mathsf{tp}}\ell$. The result is thus proved by an easy by induction on the length of $\lambda\mu\epsilon_v$-reduction sequence. ■

In addition to the fact that the previous result is fairly weak and can hardly be strengthened, we moved farther from de Groote's original calculus by introducing $\lambda\mu\epsilon_v$, while $\lambda\mu\epsilon$ was the calculus we wanted to analyse.

The direction we shall follow in the next section is on the contrary to extract a variant of $\lambda\mu\widehat{\mathsf{tp}}$ which indeed equationally corresponds to $\lambda\mu\epsilon$.

4.4 A $\lambda\mu\widehat{\mathsf{tp}}$-calculus in equational correspondence with $\lambda\mu\epsilon$

Thanks to the previous remarks, a $\lambda\mu\widehat{\mathsf{tp}}$-calculus that would be in equational correspondence with $\lambda\mu\epsilon$ would require:

- to have rule μ_{var} and not only μ^n_{var};
- redexes not to be blocked by toplevel continuation variables as in $\lambda\mu\widehat{\mathsf{tp}}\ell$ because of $\eta^v_{\widehat{\mathsf{tp}}}$.

$$
\begin{array}{lll}
(\rightarrow) & (\lambda x.\, M)\ N \longrightarrow M\{N/x\} & \\
(\mu_{\rightarrow}) & (\mu\alpha.\, c)\ M \longrightarrow \mu\beta.c\{[\beta](P)M/[\alpha]P\} & \text{if } \beta \notin FV(c, M) \\
(\mu'_{var}) & \mu\alpha.[q]\mu\beta.c \longrightarrow \mu\alpha.c\{q/\beta\} & \\
(\eta_{\mu}) & \mu\alpha.[\alpha]M \longrightarrow M & \text{if } \alpha \text{ is not free in } M \\
(\mu_{\widehat{\mathsf{tp}}}) & \left[\widehat{\mathsf{tp}}\right]\mu\widehat{\mathsf{tp}}.c \longrightarrow c & \\
(\eta_{\widehat{\mathsf{tp}}}) & \mu\widehat{\mathsf{tp}}.\left[\widehat{\mathsf{tp}}\right]M \longrightarrow M & \text{even if } \widehat{\mathsf{tp}} \text{ occurs free in } M
\end{array}
$$

Figure 17. $\lambda\mu\widehat{\mathsf{tp}}\epsilon$-reduction rules.

This suggests to consider both μ_{var} and $\eta_{\widehat{\mathsf{tp}}}$, but this cannot be done directly since these rules create a critical pair:

$$\mu\alpha.c \quad \longleftarrow_{\eta_{\widehat{\mathsf{tp}}}} \quad \mu\widehat{\mathsf{tp}}.\left[\widehat{\mathsf{tp}}\right]\mu\alpha.c \quad \longrightarrow_{\mu_{var}} \quad \mu\widehat{\mathsf{tp}}.c\left\{\widehat{\mathsf{tp}}/\alpha\right\}$$

Since we just liberalised $\eta_{\widehat{\mathsf{tp}}}$ with respect to $\lambda\mu\widehat{\mathsf{tp}}\ell$, the sensible choice to avoid the critical pair above is to forbid the right-most reduction, by requiring μ_{var} to occur only under a $\mu\alpha$ prefix and not under a $\mu\widehat{\mathsf{tp}}$ prefix. This leads to the following form of μ_{var} reduction [21] :

$$\mu\alpha.[q]\mu\beta.c \quad \longrightarrow_{\mu'_{var}} \quad \mu\alpha.c\{q/\beta\}$$

We can thus present the reduction rules of the new calculus, that we shall refer to as $\lambda\mu\widehat{\mathsf{tp}}\epsilon$:

Definition 18 *$\lambda\mu\widehat{tp}\epsilon$-calculus is defined on $\Sigma_{\lambda\mu\widehat{tp}}$ and its reduction rules are defined in figure 17.*

$\lambda\mu\widehat{\mathsf{tp}}\epsilon$ is an intermediate calculus between $\lambda\mu\widehat{\mathsf{tp}}$ and $\lambda\mu\widehat{\mathsf{tp}}\ell$ as presented in Figure 18.

We can now formulate that $\lambda\mu\epsilon$ and $\lambda\mu\widehat{\mathsf{tp}}\epsilon$ are equationally equivalent:

Proposition 19 (Equational correspondence between $\lambda\mu\epsilon$ and $\lambda\mu\widehat{\mathsf{tp}}\epsilon$) *Let $M, N \in \Sigma_{\Lambda\mu}$ and $M', N' \in \Sigma_{\lambda\mu\widehat{tp}}$. Then one has:*

- *If $M =_{\lambda\mu\epsilon} N$ then $\Pi(M) =_{\lambda\mu\widehat{tp}\epsilon} \Pi(N)$;*
- *If $M' =_{\lambda\mu\widehat{tp}\epsilon} N'$ then $\Sigma(M') =_{\lambda\mu\epsilon} \Sigma(N')$;*

[21] Notice that the formulation of μ^n_{var} on $\lambda\mu\widehat{tp}$-*terms* was $\mu q.[\alpha]\mu\beta.c \longrightarrow_{\mu^n_{var}} \mu q.c\{\alpha/\beta\}$.

$\lambda\mu\widehat{\mathsf{tp}}$	$\lambda\mu\widehat{\mathsf{tp}}\epsilon$	$\lambda\mu\widehat{\mathsf{tp}}\ell$
Syntax		
$\Sigma_{\lambda\mu\widehat{\mathsf{tp}}} \quad \ni \quad M, N ::= x \mid \lambda x.\, M \mid (M)M \mid \mu\, q.c$		
$c ::= [q]M$		
$q ::= \alpha \mid \widehat{\mathsf{tp}}$		
		$V ::= \lambda x.M \mid \mu\widehat{\mathsf{tp}}.c$
Reduction Rules		
$(\rightarrow)\ \ (\lambda x.\, M)\ N \longrightarrow M\,\{N/x\}$		
$(\mu_{\rightarrow})\ (\mu\alpha.\, c)\ M \ \longrightarrow \mu\beta.c\,\{[\beta](P)M/[\alpha]P\}$		
$(\mu^{n}_{var})\ \mu q.[\alpha]\mu\beta.c \longrightarrow \mu q.c\,\{\alpha/\beta\}$	$(\mu'_{var})\ \mu\alpha.[q]\mu\beta.c \longrightarrow \mu\alpha.c\,\{q/\beta\}$	$(\mu_{var})\ \mu q.[q']\mu\alpha.c \longrightarrow \mu q.c\,\{q'/\alpha\}$
$(\mu_{\widehat{\mathsf{tp}}})\ \left[\widehat{\mathsf{tp}}\right]\mu\widehat{\mathsf{tp}}.c \longrightarrow c$		
$(\eta_{\mu})\ \ \mu\alpha.[\alpha]M \longrightarrow M\ (\star)$		
$(\eta_{\widehat{\mathsf{tp}}})\ \mu\widehat{\mathsf{tp}}.[\widehat{\mathsf{tp}}]M \longrightarrow M\ (\star\star)$		$(\eta^{v}_{\widehat{\mathsf{tp}}})\ \mu\widehat{\mathsf{tp}}.[\widehat{\mathsf{tp}}]V \longrightarrow V\ (\star\star\star)$

($\star$) if $\alpha \notin FV(M)$
($\star\star$) (even if $\widehat{\mathsf{tp}} \in FV(M)$)
($\star\star\star$) (even if $\widehat{\mathsf{tp}} \in FV(V)$).

Figure 18. Three $\lambda\mu\widehat{\mathsf{tp}}$-calculi.

- $\Sigma(\Pi(M)) = M$;
- $\Pi(\Sigma(M')) =_{\mu_{\widehat{\mathsf{tp}}}} M'$.

PROOF: First, one shall notice the following substitution result: if $M \in \Sigma_{\Lambda\mu}$, β a continuation variable, then $\Pi(|M|_\beta) =_{\eta_{\widehat{\mathsf{tp}}}} \Pi(M)\left\{\widehat{\mathsf{tp}}/\beta\right\}$. Indeed, for every free occurrence of β in M, of the form $[\beta]N$, this is translated into $\mu\widehat{\mathsf{tp}}.[\beta]\Pi(N)$ and thus by substituting $\widehat{\mathsf{tp}}$ for β, one obtains a subterm of the form $\mu\widehat{\mathsf{tp}}.\left[\widehat{\mathsf{tp}}\right](\Pi(N)\left\{\widehat{\mathsf{tp}}/\beta\right\})$ which is $\eta_{\widehat{\mathsf{tp}}}$-equivalent to $\Pi(N)\left\{\widehat{\mathsf{tp}}/\beta\right\}$ and thus, by induction, to $\Pi(|N|_\beta)$.

We show some of the cases for the first assertion of the proposition, the other ones being non-problematic.

- case of ϵ-reduction:

$$\begin{aligned}
\Pi(\mu\alpha.\mu\beta.M) = &\quad \mu\alpha.\left[\widehat{\mathsf{tp}}\right]\mu\beta.\left[\widehat{\mathsf{tp}}\right]\Pi(M)\\
=_{\mu'_{var}} &\quad \mu\alpha.\left[\widehat{\mathsf{tp}}\right]\Pi(M)\left\{\widehat{\mathsf{tp}}/\beta\right\}\\
=_{\eta_{\widehat{\mathsf{tp}}}} &\quad \mu\alpha.\left[\widehat{\mathsf{tp}}\right]\Pi(|M|_\beta)\\
= &\quad \Pi(\mu\alpha.\,|M|_\beta)
\end{aligned}$$

- case of ρ-reduction:

$$\begin{aligned}
\Pi(\mu\alpha.[\beta]\mu\gamma.M) = &\quad \mu\alpha.\left[\widehat{\mathsf{tp}}\right]\mu\widehat{\mathsf{tp}}.[\beta]\mu\gamma.\left[\widehat{\mathsf{tp}}\right]\Pi(M)\\
=_{\mu_{\widehat{\mathsf{tp}}}} &\quad \mu\alpha.[\beta]\mu\gamma.\left[\widehat{\mathsf{tp}}\right]\Pi(M)\\
=_{\mu'_{var}} &\quad \mu\alpha.\left[\widehat{\mathsf{tp}}\right]\Pi(M)\,\{\beta/\gamma\}\\
= &\quad \Pi(\mu\alpha.M\,\{\beta/\gamma\})
\end{aligned}$$

The second part of the proposition, namely that if $M', N' \in \Sigma_{\lambda\mu\widehat{\mathsf{tp}}}$ and $M' =_{\lambda\mu\widehat{\mathsf{tp}}\epsilon} N'$ then $\Sigma(M') =_{\lambda\mu\epsilon} \Sigma(N')$, is easily checked, as well as the stability by translations (which is identical to proposition 15 since the translations are the same and both $\lambda\mu\widehat{\mathsf{tp}}$ and $\lambda\mu\widehat{\mathsf{tp}}\epsilon$ share the same $(\mu_{\widehat{\mathsf{tp}}})$ rule.

■

An interesting remark about the previous equational correspondence is that both reductions ϵ and ρ in $\lambda\mu\epsilon$ are in correspondence with μ'_{var}: the first correspondence occurs when q is $\widehat{\mathsf{tp}}$ and the second when q is a usual continuation variable.

Actually, the restriction we imposed on (μ'_{var}) plays a specific role in each of those cases: in the first case (reduction ϵ in $\lambda\mu\epsilon$), the restriction on (μ'_{var}) corresponds to the fact that ϵ will not be propagated to subterms where it

should not be, while in the second case (reduction ρ in $\lambda\mu\epsilon$), the restriction on (μ'_{var}) corresponds to the restriction on ρ in $\lambda\mu\epsilon$.

5 Conclusion

Studies on $\lambda\mu$-calculus have been numerous since Parigot introduced his calculus in 1992 [Par92], short after Griffin showed that control operators in $\lambda_{\mathcal{C}}$-calculus could be typed using axioms of classical logic [Gri90]. Many variants of Parigot's calculus arose from these various studies, which are mainly organised around two variants of the syntax that we refer to as the original syntax (or $\Sigma_{\lambda\mu}$) and the modified syntax (or $\Sigma_{\Lambda\mu}$). While the difference between the calculi had been exemplified by the separation property which holds in $\Lambda\mu$-calculus but not in $\lambda\mu$-calculus [Sau05], precise comparisons of these calculi had not been pursued [22].

Our primary goal in this paper was to give a detailed historical account of the development of $\lambda\mu$-calculus. The presentation of $\lambda\mu$-calculi, in the form of a survey, focused on the calculi built along the two variants of the syntax of $\lambda\mu$-calculus: calculi in Parigot's original syntax which respects the $\mu[]$ constraint and calculi in the modified syntax for which no such constraint is imposed. In the following two sections, we developed an analysis of the different variants of $\lambda\mu$-calculus which uses the mediation of calculi with an explicit representation of the toplevel, first in the typed setting, second in the untyped setting.

In the typed setting, we saw that the calculi in the original and the modified syntax are essentially identical as soon as one adds explicitly the $\bot$ connective as well as extensionality rules. In this case, the toplevel is represented as a *constant*, tp.

The analysis in the untyped case is more complex and subtile: in this setting, our results about $\lambda\mu\epsilon$-calculus extend previously known results connecting $\Lambda\mu$-calculus with $\lambda\mu\widehat{\mathsf{tp}}$-calculus, where the toplevel continuation is represented as a *dynamically bound variable* $\widehat{\mathsf{tp}}$. In particular, we believe our contribution emphasizes how $\lambda\mu\widehat{\mathsf{tp}}$-calculus provides an adequate framework for investigating uniformly different variants of $\lambda\mu$-calculi.

[22] Except for some results on the comparison between $\lambda\mu$-calculi that have been reported in [Sau08a] but that were much less developed than in the present article and which do not use the same techniques.

References

[AH03] Zena M. Ariola and Hugo Herbelin. Minimal classical logic and control operators. In *Thirtieth International Colloquium on Automata, Languages and Programming , ICALP'03, Eindhoven, The Netherlands, June 30 - July 4, 2003*, volume 2719, pages 871–885. Springer-Verlag, LNCS, 2003.

[AH07] Zena M. Ariola and Hugo Herbelin. Control reduction theories: the benefit of structural substitution. *J. Funct. Program.*, 2007. Includes a Historical Note by Matthias Felleisen. To appear.

[AHS04] Zena M. Ariola, Hugo Herbelin, and Amr Sabry. A type-theoretic foundation of continuations and prompts. In *Proceedings of the Ninth ACM SIGPLAN International Conference on Functional Programming, ICFP 2004, Snowbird, UT, USA, September 19-21, 2004*, pages 40–53. ACM Press, New York, 2004.

[AHS07a] Zena M. Ariola, Hugo Herbelin, and Amr Sabry. A proof-theoretic foundation of abortive continuations. *Higher Order and Symbolic Computation*, 2007. To appear.

[AHS07b] Zena M. Ariola, Hugo Herbelin, and Amr Sabry. A type-theoretic foundation of delimited continuations. *Higher Order and Symbolic Computation*, 2007. To appear.

[BB96] Franco Barbanera and Stefano Berardi. A symmetric λ-calculus for classical program extraction. *Information and Computation*, 125(2):103–117, 1996.

[CH00] Pierre-Louis Curien and Hugo Herbelin. The duality of computation. In *Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming, ICFP 2000, Montreal, Canada, September 18-21, 2000*, SIGPLAN Notices 35(9), pages 233–243. ACM, 2000.

[dG94] Philippe de Groote. On the relation between the $\lambda\mu$-calculus and the syntactic theory of sequential control. In F. Pfenning, editor, *Logic Programming and Automated Reasoning, Proc. of the 5th International Conference, LPAR'94*, volume 822 of *Lecture Notes in Artificial Intelligence*, pages 31–43. Springer-Verlag, 1994.

[dG98] Philippe de Groote. An environment machine for the $\lambda\mu$-calculus. *Mathematical Structures in Computer Science*, 8(6):637–669, 1998.

[DP01] René David and Walter Py. $\lambda\mu$-calculus and Böhm's theorem. *J. Symb. Log.*, 66(1):407–413, 2001.

[FF86] Matthias Felleisen and Daniel Friedman. Control operators, the secd machine, and the lambda-calculus. In *Formal description of programming concepts-III*, pages 193–217. North-Holland, 1986.

[FFKD86] Matthias Felleisen, Daniel P. Friedman, Eugene Kohlbecker, and Bruce F. Duba. Reasoning with continuations. In *First Symposium on Logic and Computer Science*, pages 131–141, 1986.

[Fuj03] Ken-etsu Fujita. A sound and complete cps-translation for $\lambda\mu$-calculus. In *TLCA*, pages 120–134, 2003.

[Gir91] Jean-Yves Girard. A new constructive logic: Classical logic. *Mathematical Structures in Computer Science*, 1(3):255–296, 1991.

[Gri90] Timothy G. Griffin. The formulae-as-types notion of control. In *Conf. Record 17th Annual ACM Symp. on Principles of Programming Languages, POPL '90, San Francisco, CA, USA, 17-19 Jan 1990*, pages 47–57. ACM Press, New York, 1990.

[Her05] Hugo Herbelin. *C'est maintenant qu'on calcule: au cœur de la dualité*. Habilitation à diriger les recherches, Université Paris 11, December 2005.

[HG08] Hugo Herbelin and Silvia Ghilezan. An approach to call-by-name delimited continuations. In *POPL*. ACM Sigplan, January 2008.

[HGS09] Hugo Herbelin, Silvia Ghilezan, and Alexis Saurin. A uniform approach to call-by-name and call-by-value delimited control. manuscript, 2009.

[Hof95] Martin Hofmann. Sound and complete axiomatisations of call-by-value control operators. *Mathematical Structures in Computer Science*, 5(4):461–482, 1995.

[HS97] Martin Hofmann and Thomas Streicher. Continuation models are universal for $\lambda\mu$-calculus. In *LICS*, pages 387–395, 1997.

[HS02] Martin Hofmann and Thomas Streicher. Completeness of continuation models for $\lambda\mu$-calculus. *Inf. Comput.*, 179(2):332–355, 2002.

[Kri93] Jean-Louis Krivine. *Lambda-calculus, types and models*. Ellis Horwood, 1993.

[LRS93] Yves Lafont, Bernhard Reus, and Thomas Streicher. Continuations semantics or expressing implication by negation. Technical Report 9321, Ludwig-Maximilians-Universität, München, 1993.

[Mur91] Chetan R. Murthy. An evaluation semantics for classical proofs. In *Proceedings, Sixth Annual IEEE Symposium on Logic in Computer Science, 15-18 July, 1991, Amsterdam, The Netherlands*, pages 96–107. IEEE Computer Society, 1991.

[Mur92] Chetan R. Murthy. A computational analysis of girard's translation and lc. In *Proc. of the Seventh Annual IEEE Symposium on Logic in Computer Science*, pages 90–101, Santa Cruz, CA, 1992.

[Ong96] C.-H. Luke Ong. A semantic view of classical proofs: type-theoretic, categorical, denotational characterizations. In *Proceedings of 11th IEEE Annual Symposium on Logic in Computer Science*, pages 230–241. IEEE Computer Society Press, 1996.

[OS97] C.-H. Luke Ong and Charles A. Stewart. A Curry-Howard foundation for functional computation with control. In *Proceedings of ACM SIGPLAN-SIGACT Symposium on Principle of Programming Languages, Paris, January 1997*, pages 215–227. ACM Press, 1997.

[Par91] Michel Parigot. Free deduction: An analysis of Computations in classical logic. In Andrei Voronkov, editor, *Logic Programming, First Russian Conference on Logic Programming, Irkutsk, Russia, September 14-18, 1990 - Second Russian Conference on Logic Programming, St. Petersburg, Russia, September 11-16, 1991, Proceedings*, volume 592 of *Lecture Notes in Computer Science*, pages 361–380. Springer, 1991.

[Par92] Michel Parigot. $\lambda\mu$-calculus: An algorithmic interpretation of classical natural deduction. In *Logic Programming and Automated Reasoning: International Conference LPAR '92 Proceedings, St. Petersburg, Russia*, pages 190–201. Springer-Verlag, 1992.

[Par93] Michel Parigot. Classical proofs as programs. In *Proceedings of the Third Kurt Gödel Colloquium on Computational Logic and Proof Theory*, volume 713 of *LNCS*, pages 263 – 276, London, UK, 1993. Springer-Verlag.

[Py98] Walter Py. *Confluence en $\lambda\mu$-calcul*. Thèse de doctorat, Université de Savoie, 1998.

[RS94] Jakob Rehof and Morten Heine Sørensen. The λ_Δ-calculus. In Masami Hagiya and John C. Mitchell, editors, *Theoretical Aspects of Computer Software, International Conference TACS '94, Sendai, Japan, April 19-22, 1994, Proceedings*, volume 789 of *Lecture Notes in Computer Science*, pages 516–542. Springer, 1994.

[Sau05] Alexis Saurin. Separation with streams in the $\Lambda\mu$-calculus. In *Proceedings, 20th Annual IEEE Symposium on Logic in Computer Science (LICS '05)*, pages 356–365. IEEE Computer Society Press, 2005.

[Sau08a] Alexis Saurin. On the relations between the syntactic theories of $\lambda\mu$-calculi. In *17th EACSL Annual Conference on Computer Science Logic 2008 (CSL 2008)*, LNCS. Springer, September 2008.

[Sau08b] Alexis Saurin. *Une étude logique du contrôle, appliquée à la programmation fonctionnelle et logique*. PhD thesis, École Polytechnique, September 2008.

[Sel01] Peter Selinger. Control categories and duality: on the categorical semantics of the lambda-mu calculus. *Mathematical Structures in Computer Science*, 11(2):207–260, 2001.

[SF93] Amr Sabry and Matthias Felleisen. Reasoning about programs in continuation-passing style. *Lisp and Symbolic Computation*, 6(3-4):289–360, 1993.

[SR98] Thomas Streicher and Bernhard Reus. Classical logic, continuation semantics and abstract machines. *J. Funct. Program.*, 8(6):543–572, 1998.