Towards a canonical classical natural deduction system

José Espírito Santo
Centro de Matemática
Universidade do Minho
Portugal
jes@math.uminho.pt

Abstract

This paper studies a new classical natural deduction system, presented as a typed calculus named $\underline{\lambda}\mu\mathsf{let}$. It is designed to be isomorphic to Curien and Herbelin's $\overline{\lambda}\mu\tilde{\mu}$-calculus, both at the level of proofs and reduction, and the isomorphism is based on the correct correspondence between cut (resp. left-introduction) in sequent calculus, and substitution (resp. elimination) in natural deduction. It is a combination of Parigot's $\lambda\mu$-calculus with the idea of "coercion calculus" due to Cervesato and Pfenning, accommodating let-expressions in a surprising way: they expand Parigot's syntactic class of named terms.

This calculus and the mentioned isomorphism Θ offer three missing components of the proof theory of classical logic: a canonical natural deduction system; a robust process of "read-back" of calculi in the sequent calculus format into natural deduction syntax; a formalization of the usual semantics of the $\overline{\lambda}\mu\tilde{\mu}$-calculus, that explains co-terms and cuts as, respectively, contexts and hole-filling instructions. $\underline{\lambda}\mu\mathsf{let}$ is not yet another classical calculus, but rather a canonical reflection in natural deduction of the impeccable treatment of classical logic by sequent calculus; and Θ provides the "read-back" map and the formalized semantics, based on the precise notions of context and "hole-expression" provided by $\underline{\lambda}\mu\mathsf{let}$.

We use "read-back" to achieve a precise connection with Parigot's $\lambda\mu$, and to derive λ-calculi for call-by-value combining control and let-expressions in a logically founded way. Finally, the semantics Θ, when fully developed, can be inverted at each syntactic category. This development gives us license to see sequent calculus as the semantics of natural deduction; and uncovers a new syntactic concept in $\overline{\lambda}\mu\tilde{\mu}$ ("co-context"), with which one can give a new definition of η-reduction.

Contents

1. Introduction

In this paper we introduce a new natural deduction system $\underline{\lambda}\mu\mathsf{let}$ for classical logic, presented as an extension of Parigot's $\lambda\mu$-calculus [25], and equipped with let-expressions. The main design principle of $\underline{\lambda}\mu\mathsf{let}$ was to obtain a system *isomorphic* to the classical sequent calculus $\overline{\lambda}\mu\tilde{\mu}$ of Curien and Herbelin [7], in order to have, in the natural deduction side, a system as faithful to the classical dualities as $\overline{\lambda}\mu\tilde{\mu}$. For this reason, $\underline{\lambda}\mu\mathsf{let}$ is not yet another calculus for classical logic, but rather a canonical reflection in natural deduction of the impeccable treatment of classical logic by sequent calculus.

The design of $\underline{\lambda}\mu\mathsf{let}$ is not a mere formal achievement; we will try to prove that $\underline{\lambda}\mu\mathsf{let}$ is full of syntactic subtlety, semantic insight, and, mainly, that it is the appropriate tool to make progress simultaneously in three different but related areas: (i) semantics of $\overline{\lambda}\mu\tilde{\mu}$; (ii) CBV λ-calculus; (ii) natural deduction for classical logic. Before we explain $\underline{\lambda}\mu\mathsf{let}$ in more detail, we expand on the problems we will address in these three areas.

Semantics of $\overline{\lambda}\mu\tilde{\mu}$. Gentzen [15] refined the de Morgan classical duality from the level of provability to the level of proofs, by defining the sequent calculus LK, a symmetric proof system for classical logic exhibiting a duality between hypothesis and conclusion. Recently Curien and Herbelin [7] introduced a variant of LK and the corresponding $\overline{\lambda}\mu\tilde{\mu}$-calculus, extending the Curry-Howard correspondence to classical sequent calculus, and showing that classical logic also contains a duality, at the level of cut elimination, between call-by-name (CBN) and call-by-value (CBV) computation.

Even if it is clear that $\overline{\lambda}\mu\tilde{\mu}$ is some sort of functional language with control facilities, its full understanding rests, so far, on intuitions that are vague and deserve to be formalized. We mean the explanation of co-terms as "contexts" (in particular, the $\tilde{\mu}$ operator is explained in terms of a let-expression with a hole); and the explanation of cuts as "hole filling" in those contexts [7, 19]. These contexts are derived from some natural deduction syntax, some variant of $\lambda\mu$, extended with let-expressions. But which variant exactly? We prove the answer is $\underline{\lambda}\mu\mathsf{let}$. The contexts that interpret co-terms are a derived syntactic notion of $\underline{\lambda}\mu\mathsf{let}$; and filling the hole of such contexts results in expressions of a certain syntactic class of $\underline{\lambda}\mu\mathsf{let}$ named *statements*. This semantics is nothing less than the isomorphism $\Theta : \overline{\lambda}\mu\tilde{\mu} \to \underline{\lambda}\mu\mathsf{let}$.[1]

The CBV λ-calculus. Through $\overline{\lambda}\mu\tilde{\mu}$, Curien and Herbelin reduced the essence of the non-determinism in classical cut-elimination to a single critical pair, and recognized in this critical pair the choice between CBN and CBV computation. In particular, this opened the way to the definition of CBV fragments of $\overline{\lambda}\mu\tilde{\mu}$ and to a proof-theoretical answer to the question "what is CBV λ-calculus?", a question firstly posed in [26], and explicitly addressed in [7]. Such proof-theoretical approach contrasts with the developments in [22, 32, 33] that put forward Moggi's computational λ-calculus as the CBV λ-calculus.

For a variety of reasons, one would like to see the sequent calculus account of CBV translated to natural deduction. First, because that would provide a "read-back"[7, 19] of $\overline{\lambda}\mu\tilde{\mu}$ proof expressions into a language where the familiar notation of functional application is available and, therefore, a language closer to actual programming languages. Second, because it is rather natural to ask whether the proof-theoretical understanding of CBV is an exclusive of sequent calculus, that is, whether natural deduction is, for some reason, doomed to account only for CBN computation. This read-back effort, already found in [7], continued through [19, 30, 20]. But the effort shows many difficulties. It is to a large extent informal, as the target system is not properly developed; it shows hesitations, as some attempts admittedly failed [19]; and it is even contradictory, as the definition of CBV $\lambda\mu$-calculus in [30] disagrees with that of [24].

We propose the isomorphism $\Theta : \overline{\lambda}\mu\tilde{\mu} \to \underline{\lambda}\mu\mathsf{let}$ as a systematic read-back process, with

[1]When the reader sees $\underline{\lambda}\mu\mathsf{let}$ maybe (s)he will be surprised: how can such a system be the "explanation" of $\overline{\lambda}\mu\tilde{\mu}$? The measure of the reader's surprise is the measure of how vague and unclear were for her/him the explanations in terms of contexts and hole filling, and of how much the formalization of such explanations is needed.

a fully formalized and developed natural deduction target. In particular, one obtains CBV λ-calculi in natural deduction format by restricting Θ to CBV fragments of $\overline{\lambda}\mu\tilde{\mu}$ and characterizing the range of such restrictions.

Natural deduction for classical logic. As we are seeing, the need to develop natural deduction for classical logic has many external motivations, but it also arises from the internal difficulties of the theory of natural deduction.

Both Gentzen [15] and Prawitz [28] defined natural deduction for classical logic as intuitionistic natural deduction supplemented with some classical inference principle. Prawitz admits that "this is perhaps not the most natural procedure from the classical logic point of view", as it does not reflect the de Morgan symmetry at the level of proofs ([28], pg.44); and Gentzen observed that there is no canonical choice as to what inference principle to add. Computationally, through the Curry-Howard correspondence, this means that the λ-calculus may be extended with a variety of control operators: for instance $\mathcal{C}$, Δ, or `call-cc`, corresponding to the principles double-negation elimination, *reductio ad absurdum*, and Peirce's law, respectively [14, 17, 29, 1].

So, nothing like a canonical system is obtained through Gentzen-Prawitz approach to classical natural deduction. Moreover, the design difficulties are accompanied by technical problems. According to [35], Prawitz [28] proves only a "slightly weakened subformula property"; and restriction of *reductio ad absurdum* to atomic conclusions only works for some logical constants [28, 34]. The latter problem is solved in [23] through the adoption of general elimination rules [36] and replacing *reductio ad absurdum* with another principle: elimination from excluded middle. A "structural" approach to overcome the mentioned difficulties is to move to systems whose deductions conclude not a single formula, but rather a list or set of formulas. This is already found in the system of Boričić [5], where an elegant subformula property holds. But the full computational power of the explicit manipulation of several conclusions is revealed by Parigot in his $\lambda\mu$-calculus [25].

Adopting the multiple conclusion framework allows us to depart from the intuitionistic system in a way that avoids having to chose among the variety of classical principles. But it still does not guarantee a natural deduction system faithful to the classical dualities. This is easy to verify: by Curien and Herbelin, the duality CBN/CBV is a duality of classical logic; but it is not clear even how to define a CBV variant of Parigot's $\lambda\mu$, as the distinct proposals [24, 7, 19, 30, 20] show, let alone give a natural deduction explanation of CBV based on $\lambda\mu$. One of the problems is that let-expressions are unavoidable in CBV λ-calculi, and therefore one has to offer some proof-theoretical understanding of them. In particular, natural deduction has to be extended. But how? This paper claims that let-expressions *cannot* be added to $\lambda\mu$ in a routine way, taking them as another form of proof-term; moreover, one has to understand the difference between the typing rule for let-expressions (which is a substitution inference rule) and the cut rule in sequent calculus. Cut and substitution are (perhaps in a subtle way) different, although linked by the isomorphism $\Theta : \overline{\lambda}\mu\tilde{\mu} \rightarrow \underline{\lambda}\mu\mathsf{let}$.

The new system. $\underline{\lambda}\mu\mathsf{let}$ is an extension with let-expressions of Parigot's $\lambda\mu$-calculus; in addition, it is a "coercion calculus" [6, 10, 12], with its syntax carefully organized into three syntactic classes. Besides the class of terms, it has a class of *statements*, extending Parigot's named terms - surprisingly, this is where let-expressions live; and also a class of *hole expressions* suitable to be filled in the hole of *contexts*. Contexts are a derived syntactic class and consist of statements with a single hole in the left end. Applications are hole expressions, not terms, and have the form HN, with H a hole expression and N a term. Like in sequent calculus, in $\underline{\lambda}\mu\mathsf{let}$ ordinary application of two terms has to be derived.

As a proof system, $\underline{\lambda}\mu\mathsf{let}$ extends Parigot's typing system for $\lambda\mu$, sometimes called "classical natural deduction". In particular, the power of classical logic is achieved through the device of activating or making passive one of the multiple conclusions. Parigot's typing system manipulates two kinds of sequent (corresponding to terms and named term), whereas $\underline{\lambda}\mu\mathsf{let}$ manipulates three. The novelty is in the sequents corresponding to hole expressions, which play an interesting role in the proof of the subformula property.

$\underline{\lambda}\mu\mathsf{let}$ has a *substitution* inference rule[2] which is the typing rule for the let-expressions constructor, and therefore its Curry-Howard counterpart. However, one has to argue why the substitution inference rule is a natural deduction rule. Substitution is different from cut because: (i) its left premiss can be the conclusion of an elimination (while cut's left premiss can't); (ii) primitive substitution's right premiss can't be the conclusion of a left-introduction (while cut's right premiss can). This is, in sketchy terms, the proof-theoretical understanding of let-expressions put forward by this paper.

Finally, $\underline{\lambda}\mu\mathsf{let}$ is equipped with reduction rules which correspond to a normalization procedure. We also consider the natural deduction rules corresponding to the η-like rules η_μ and $\eta_{\tilde{\mu}}$ of $\overline{\lambda}\mu\tilde{\mu}$. The normalization rules have redexes of the form $\mathcal{E}[H]$, where $\mathcal{E}$ is a context. This looks complex at first sight, but inevitable after a second thought. $\underline{\lambda}\mu\mathsf{let}$ combines control operation with "superimposed" [19] CBN and CBV functional computation. Explicit manipulation of contexts in reduction rules is inherent to both CBV λ-calculi and calculi with control operators [14, 17, 32, 24, 20].

The normalization rules work roughly as follows. Take a redex $\mathcal{E}[H]$. H may be an ordinary β-redex or a μ-abstraction. In the first case, the reduction produces a let-expression; and if this let-expression is in a particular form corresponding to a delayed substitution, a separate reduction rule may fire the corresponding meta-substitution. In the second case, the reduction fires a suitable meta-substitution $[\mathcal{E}/a]\_$.

The isomorphism. The isomorphism $\Theta : \overline{\lambda}\mu\tilde{\mu} \to \underline{\lambda}\mu\mathsf{let}$ is the only and full justification of why $\underline{\lambda}\mu\mathsf{let}$ is the way it is (just to make it clear, Θ comprises a sound bijection at the level of proof expressions, and an isomorphism at the level of reduction relations). The isomorphism has many facets (natural deduction semantics, read-back process), but its foremost aspect is proof-theoretical: the central idea is to replace one left-introduction inference by a corresponding elimination inference. Computationally, this corresponds to the "inversion of associativity" of applicative terms (=non-values), an idea that Herbelin [18] recognized to be Curry-Howard explanation of the difference between sequent calculus and natural deduction. The author has developed this idea in the intuitionistic case since [10], recognizing in it the design principle for the expansion of the natural deduction realm. The present paper can be seen as a passage to the classical case of results in [12].

[2]In order to allow ourselves some relief from the multiple uses of the word "substitution" (see Subsection 2.1), we might call the substitution inference rule the "let inference rule", or even *let* and speak of lets as we all speak of cuts.

Structure of the paper. The paper is organized as follows. Section 2 fixes notation and syntactic conventions, and recalls $\lambda\mu$ and $\overline{\lambda}\mu\tilde{\mu}$. Section 3 presents $\underline{\lambda}\mu\mathsf{let}$, proves the subformula property and discusses the system. Section 4 proves the isomorphism $\overline{\lambda}\mu\tilde{\mu} \cong \underline{\lambda}\mu\mathsf{let}$ and explains the proof-theoretical foundation of let-expressions, delayed substitutions and named terms. Section 5 investigates CBN and CBV in $\underline{\lambda}\mu\mathsf{let}$, and the connections of $\lambda\mu$ with $\underline{\lambda}\mu\mathsf{let}$. Section 6 studies the natural deduction semantics of $\overline{\lambda}\mu\tilde{\mu}$ offered by $\underline{\lambda}\mu\mathsf{let}$. Section 7 summarizes the paper, reviews the literature, and suggests future work. Appendix A recalls the CBN and CBV fragments of $\overline{\lambda}\mu\tilde{\mu}$.

This paper is a revised and substantially expanded version of the conference paper [13]. Proofs are included, the presentation is improved, and motivations for studying $\underline{\lambda}\mu\mathsf{let}$ are detailed. New material consists of: full treatment of the CBN and CBV fragments of $\underline{\lambda}\mu\mathsf{let}$; study of the relationship between $\lambda\mu$ and $\underline{\lambda}\mu\mathsf{let}$; full development of the semantics of $\underline{\lambda}\mu\mathsf{let}$, which requires a specific treatment of contexts in $\underline{\lambda}\mu\mathsf{let}$ and a new concept in $\overline{\lambda}\mu\tilde{\mu}$ called "co-context". Incidentally, the latter permits a new definition of η-reduction in $\overline{\lambda}\mu\tilde{\mu}$.

2. Background

In this section we clarify some syntactic matters and options (notation, overloading, variable convention, substitution, derived syntax, and abbreviations), and recall Parigot's $\lambda\mu$ [25] and Curien and Herbelin's $\overline{\lambda}\mu\tilde{\mu}$ [7].

2.1. Notation and other syntactic preliminaries.

Syntax of λ-calculi. This paper is about λ-calculi for classical logic, both in the format of sequent calculus ($\overline{\lambda}\mu\tilde{\mu}$) or natural deduction ($\lambda\mu$ and the new $\underline{\lambda}\mu\mathsf{let}$). Terms will be denoted t, u, v in sequent calculus, and M, N, P in natural deduction. Terms can always be separated into values and non-values. A *value* is a variable or λ-abstraction, and is always denoted V or W.

In natural deduction, if $\vec{N} = N_1, \cdots, N_m$ ($m \geq 0$), then $H\vec{N}$ denotes $HN_1 \cdots N_m$, that is, $(\cdots(HN_1)\cdots N_m))$, for H some expression. In sequent calculus, if $\vec{u} = u_1, \cdots, u_m$ ($m \geq 0$), then $\vec{u} :: e$ denotes $u_1 :: \cdots :: u_m :: e$, that is, $(u_1 :: \cdots :: (u_m :: e)\cdots))$, for some expression e. This "inversion of associativity" (natural deduction is left associative, sequent calculus is right associative) is a cornerstone of the present paper, and goes back to Herbelin [18].

In λ-calculi for classical logic there are variables and co-variables. Variables (resp. co-variables) are always ranged by x, y, z (resp. a, b, c). All calculi in this paper assume Barendregt's variable convention (in particular we take renaming of bound variables or co-variables and avoidance of capture for granted).

For each λ-calculus, a number of reduction rules is defined. Given a reduction rule R, the symbol $\to_R$ denotes the compatible closure of R, that is, the closure of R for any syntactic constructor of expressions in the calculus. The transitive (resp. reflexive-transitive) closure of $\to_R$ is denoted $\to_R^+$ (resp. $\to_R^*$).

Contexts. In the ordinary λ-calculus, a context is a λ-term with "holes" - with holes denoted $[\_]$ in this paper. In our view, the concept of λ-term is defined first, and the concept of context is defined afterwards, on top of the concept of λ-term. For instance, in the context $[\_]N_1 \cdots N_m$, we assume the set from which the N_i's are taken (to wit, the set of λ-terms) to be previously defined.

In this paper we adopt a similar view for all the calculi considered, with the exception that we are interested only in contexts with a single hole. Contexts are denoted $\mathcal{C}$, $\mathcal{E}$, etc., and the result of filling some expression H in the hole of $\mathcal{C}$ is denoted $\mathcal{C}[H]$.

Derived syntax. Is it superfluous to introduce abbreviations for easily derived syntactic concepts? Suppose we wanted to develop a rudimentary "pairing" calculus on the λ-terms. A easily derived pairing construction is $(M, N) := (\lambda z.M)N$, with $z \notin M$. Would it be worthwhile to introduce the abbreviation (M, N) above, given that it can easily be reduced to the primitive syntax of the λ-terms? In this paper we adopt the answer "yes" to such questions, and indeed we would even write the definition of the projection reduction rule as $(M, N) \to M$, simply to stress the intention of a "pairing" calculus. In the calculi considered here, similar questions arise in connection with *explicit substitution*, which will be an easily derived term constructor.

Substitution. We have to distinguish carefully three concepts of "substitution": (i) the meta-operation of substitution (or the result of such operation); (ii) explicit substitution (or its variant "delayed" substitution); the substitution typing/inference rule.

Meta-substitution is an operation defined on the set of terms and denoted with square brackets, following [3], but in the style $[\_/x]\_$ (not in the style $\_[x := \_]$). Similarly for other forms of meta-substitution used in λ-calculi for classical logic, like "structural" substitution.

Explicit substitution is a term constructor denoted with angle brackets, as in λx-calculus papers [31, 4], again in the style $\langle\_/x\rangle\_$ (not in the style $\_\langle x := \_\rangle$). In explicit substitution calculi, not only explicit substitution is a term constructor, but also reduction rules exist for the stepwise execution of substitution. An intermediate concept is that of a "delayed substitution" calculus [11], where substitution is a term constructor but only a single reduction rule exists for substitution execution, which fires and executes completely the substitution in a single step of reduction. In this paper we adopt the latter approach, as we are not interested in the stepwise execution of substitution.

A substitution typing rule is simply a typing rule for some concept of substitution, *e.g.* the following typing rule taken from the simply typed λ-calculus:

$$\frac{\Gamma \vdash N : A \quad \Gamma, x : A \vdash M : B}{\Gamma \vdash [N/x]M : B} \ .$$

A substitution typing rule can thus be an admissible typing rule if we are typing meta-substitution, or a primitive rule of the typing system as in explicit substitution calculi. From the logical point of view, one may prefer to speak of a substitution inference (rather than typing) rule. An important point of this paper is the explanation of the correspondence between substitution inference rules belonging to natural deduction systems and the cut rule of sequent calculus.

Typing. Formulas (=types) are ranged over by A, B, C and generated from type variables using implication, written $A \supset B$. There will be no symbol for absurdity, so the logic we work with is "minimal classical logic"[1]. Contexts Γ are consistent sets of declarations $x : A$. "Consistent" means that for each variable x there is at most one declaration in Γ. We assume $\Gamma, x : A$ always denotes a consistent set. Similarly for co-contexts Δ, consistent sets of declarations $a : A$ of co-variables.

Overloading. We can say that $\overline{\lambda}\mu\tilde{\mu}$, $\lambda\mu$, and the new $\underline{\lambda}\mu\mathsf{let}$ are "λ-calculi for classical logic". As such, it is most natural each of them carries its own β-reduction rule. Absolute rigor would demand to tag the name "β" to distinguish β-rules from different systems; similarly, one would tag the substitution that a β-rule always generates. However, as long as context resolves ambiguity, we will avoid tagging as much as possible, just to keep the notation lighter. The same applies, for instance, to the symbol $\vdash$.

The overloading of brackets is unavoidable, but it is reduced to two situations: (i) square brackets are used in meta-substitution and in holes of contexts; (ii) angle brackets are used in explicit/delayed substitutions and in "commands" from $\overline{\lambda}\mu\tilde{\mu}$. At least, the use of square brackets in the naming constructor of $\lambda\mu$ (and $\underline{\lambda}\mu\mathsf{let}$) was abolished: we write $a(M)$ (not $[a]M$).

Finally, overloading is quite often illuminating, simply because many syntactic concepts transcend a particular system where they may happen to be defined, and make sense across several systems. Obvious examples come from the λ-calculus: λ-abstraction, substitution, and application. The symbol λ is re-used across systems, as well as the square brackets notation for substitution. As to the concept of ordinary application of two terms, which we denote by juxtaposition MN, this concept is again primitive in the $\lambda\mu$-calculus and denoted in the same way. In the sequent calculus, ordinary application of two terms is not primitive, but can be derived; once a derivation is agreed upon (there may be several), we find it most natural and economical to re-use the juxtaposition notation and denote the derived concept by tu.

The same re-use is applied to the names of typing rules. For instance, once tu is defined in the sequent calculus in a sound way, its comes with a derived typing rule which we call "elimination". Similarly we can derive in natural deduction a left-introduction construction and corresponding inference rule. As soon as "elimination" and "left-introduction" are re-used in this way, they become inference principles in both natural deduction and sequent calculus. Then, what distinguishes the two proof systems is which of these principles are *primitive* and which are derived.

2.2. $\lambda\mu$-calculus

Expressions of $\lambda\mu$ are defined by the following grammar:

$$\begin{array}{lrcl} \text{(Terms)} & M,N,P & ::= & x \mid \lambda x.M \mid MN \mid \mu a.S \\ \text{(Named terms)} & S & ::= & a(M) \end{array}$$

The terminology "named term" comes from the fact that co-variables are sometimes called "names".

In addition to ordinary substitution for variables, there are the operators $[b/a]M$ and $[b/a]S$ for the renaming of free occurrences of a, and the *"structural" meta-substitution* operators $[a([\_]N)/a]M$ and $[a([\_]N)/a]S$, in whose recursive definition the crucial clause is:

$$[a([\_]N)/a](a(M)) = a(M'N) \text{ where } M' = [a([\_]N)/a]M \ .$$

We consider 4 reduction rules:

Figure 1: Typing rules for $\lambda\mu$

$$\frac{}{\Gamma, x : A \vdash x : A|\Delta}\ Assumption$$

$$\frac{\Gamma \vdash M : A \supset B|\Delta \quad \Gamma \vdash N : A|\Delta}{\Gamma \vdash MN : B|\Delta}\ Elim \qquad \frac{\Gamma, x : A \vdash t : B|\Delta}{\Gamma \vdash \lambda x.t : A \supset B|\Delta}\ Intro$$

$$\frac{\Gamma \vdash M : A|\Delta, a : A}{a(M) : (\Gamma \vdash \Delta, a : A)}\ Pass \qquad \frac{S : (\Gamma \vdash \Delta, a : A)}{\Gamma \vdash \mu a.S : A|\Delta}\ Act$$

$$\begin{array}{crcl}
(\beta) & (\lambda x.M)N & \to & [N/x]M \\
(\mu) & (\mu a.S)N & \to & \mu a.[a([\_]N)/a]S \\
(\eta_\mu) & \mu a.a(M) & \to & M, \text{ if } a \notin M \\
(\rho) & b(\mu a.S) & \to & [b/a]S
\end{array}$$

In the typing system there is one kind of sequent per each syntactic class $\Gamma \vdash t : A|\Delta$ and $S : (\Gamma \vdash \Delta)$. In the first kind of sequents, A is said to be *active*. The typing rules are given in Figure 1. We have the typing rules of the ordinary λ-calculus, plus an *activation* rule and a "*passivation*" rule. Proof-theoretically, this is a natural deduction system deriving sequent with multiple conclusions.

2.3. $\overline{\lambda}\mu\tilde{\mu}$-calculus

Expressions of $\overline{\lambda}\mu\tilde{\mu}$ are defined by the following grammar:

$$\begin{array}{rccl}
\text{(Terms)} & t, u & ::= & x \mid \lambda x.t \mid \mu a.c \\
\text{(Co-terms)} & e & ::= & a \mid u :: e \mid \tilde{\mu} x.c \\
\text{(Commands)} & c & ::= & \langle t|e \rangle
\end{array}$$

A *value* is a term of the form x or $\lambda x.t$. A *co-value* is a co-term of the form a or $u :: e$. That is: a value is a term not of the form $\mu a.c$ and a co-value is a co-term not of the form $\tilde{\mu} x.c$.

There is one kind of sequent per each syntactic class

$$\Gamma \vdash t : A|\Delta \qquad \Gamma|e : A \vdash \Delta \qquad c : (\Gamma \vdash \Delta)\ .$$

In the first two kinds, the displayed formula A is *active*. The typing rules are given in Figure 2. A *typable* term is a term t such that $\Gamma \vdash t : A|\Delta$ is derivable, for some Γ, Δ, A. Similarly for co-terms e and commands c.

Because of the typing rule *Cut*, we often refer to commands as cuts. Every cut has one of two forms:

$$\begin{array}{ll}
\langle t|u_1 :: \cdots :: u_m :: a \rangle & \text{(covar-cut)} \\
\langle t|u_1 :: \cdots :: u_m :: \tilde{\mu} x.c \rangle & (\tilde{\mu}\text{-cut})
\end{array}$$

Figure 2: Typing rules for $\overline{\lambda}\mu\tilde{\mu}$

$$\frac{}{\Gamma|a:A\vdash a:A,\Delta}\,LAx \qquad \frac{}{\Gamma,x:A\vdash x:A|\Delta}\,RAx$$

$$\frac{\Gamma\vdash u:A|\Delta \quad \Gamma|e:B\vdash\Delta}{\Gamma|u::e:A\supset B\vdash\Delta}\,LIntro \qquad \frac{\Gamma,x:A\vdash t:B|\Delta}{\Gamma\vdash\lambda x.t:A\supset B|\Delta}\,RIntro$$

$$\frac{c:(\Gamma,x:A\vdash\Delta)}{\Gamma|\tilde{\mu}x.c:A\vdash\Delta}\,LAct \qquad \frac{c:(\Gamma\vdash a:A,\Delta)}{\Gamma\vdash\mu a.c:A|\Delta}\,RAct$$

$$\frac{\Gamma\vdash t:A|\Delta \quad \Gamma|e:A\vdash\Delta}{\langle t|e\rangle:(\Gamma\vdash\Delta)}\,Cut$$

Figure 3: Reduction rules of $\overline{\lambda}\mu\tilde{\mu}$

$$\begin{array}{lrcll}
(\beta_\supset) & \langle\lambda x.t|u::e\rangle & \to & \langle u|\tilde{\mu}x.\langle t|e\rangle\rangle & \\
(\sigma_{\tilde{\mu}}) & \langle t|\tilde{\mu}x.c\rangle & \to & [t/x]c & \\
(\sigma_\mu) & \langle\mu a.c|e\rangle & \to & [e/a]c & \\
 & & & & \\
(\eta_\supset) & \lambda x.\mu a.\mathcal{H}[x::a] & \to & \mu a.\mathcal{H}[a] & (x,a\notin\mathcal{H}) \\
(\eta_{\tilde{\mu}}) & \tilde{\mu}x.\langle x|e\rangle & \to & e & (x\notin e) \\
(\eta_\mu) & \mu a.\langle t|a\rangle & \to & t & (a\notin t)
\end{array}$$

Proof-theoretically, the system in Figure 2 is a sequent calculus, with cut and left introduction instead of elimination. There is a right activation rule, like the activation rule of $\lambda\mu$, but also a left activation rule, that types the $\tilde{\mu}$-abstraction. Pushing the comparison with $\lambda\mu$ further, we may say that the cut rule generalises the passivation rule of $\lambda\mu$. Indeed, we might call $\langle t|a\rangle$ a named term; but in $\overline{\lambda}\mu\tilde{\mu}$ we have the dual passivation form $\langle x|e\rangle$, which we might call a dereliction, following [18].

In addition to three ordinary substitution operators $[t/x]c$, $[t/x]u$, and $[t/x]e$, there are three *co-substitution* operators $[e/a]c$, $[e/a]u$, and $[e/a]e'$.

We consider 6 reduction rules - see Figure 3[3]. By *cut-elimination* we mean $\beta_\supset\sigma_{\tilde{\mu}}\sigma_\mu$-reduction. In addition to the cut-elimination rules, there are three η-like rules. The rules η_μ and $\eta_{\tilde{\mu}}$ are considered *e.g.* in [27]. A new η-rule, named $\eta_\supset$, is proposed but its discussion is entirely postponed to Subsection 6.5.

A $\sigma_{\tilde{\mu}}$-redex (resp. σ_μ-redex) is a cut that is permutable to the right (resp. left), because its right (resp. left) cut-formula is not principal in a left (resp. right) introduction. A $\beta_\supset$-redex is a cut whose both cut-formulas are principal in introduction inferences,

[3]We follow [27] in using β to name the first rule (see Subsection 2.1 for more on giving name "β" to rules from different calculi). The reduction rules usually named $\tilde{\mu}$ and μ are here renamed $\sigma_{\tilde{\mu}}$ and σ_μ, respectively. The intention is to stress that these are rules for executing a delayed (co-)substitution, as explained below.

and of the form $A \supset B$. $\beta_\supset$-reduction breaks this cut into two cuts with cut-formulas A and B.

The $\beta_\supset\sigma_{\tilde\mu}\sigma_\mu$-normal forms are the expressions where every command has one of the forms $\langle V|a\rangle$ or $\langle x|E\rangle$. These forms are the passivation of (co-)values.

There is a critical pair

$$\begin{tikzcd}[\tilde\mu x.c'/a]c & \langle\mu a.c|\tilde\mu x.c'\rangle \arrow[l, "\sigma_\mu"'] \arrow[r, "\sigma_{\tilde\mu}"'] & [\mu a.c/x]c'\end{tikzcd} \qquad (1)$$

It is well known that this critical pair in general cannot be joined (just take the case $a \notin c$, $x \notin c'$ - this is Lafont's counter-example [16]), and so confluence fails in $\overline{\lambda}\mu\tilde\mu$. According to [7], avoiding this critical pair by giving priority to one or the other reduction is the principle for the definition of the CBN (priority to $\sigma_{\tilde\mu}$) or CBV (priority to σ_μ) reduction. For this reason, we refer to this critical pair as the *CBN-CBV dilemma.*

In $\overline{\lambda}\mu\tilde\mu$ we define:[4]

$$\begin{array}{rcll}
a(t) & := & \langle t|a\rangle & \text{(named term)}\\
tu & := & \mu b.\langle t|u :: b\rangle & \text{(ordinary application)}\\
\langle t/x\rangle c & := & \langle t|\tilde\mu x.c\rangle & \text{(delayed substitution)}\\
\langle e/a\rangle c & := & \langle \mu a.c|e\rangle & \text{(delayed co-substitution)}
\end{array}$$

In these definition, of course, the new bound (co-)variables are assumed fresh.

The first two definitions define homomorphically a sound (*i.e.* type preserving) translation of the natural deduction $\lambda\mu$ into $\overline{\lambda}\mu\tilde\mu$ (this is translation $(\_)^n$ in [7]). Another interpretation of $\lambda\mu$ into $\overline{\lambda}\mu\tilde\mu$, also given in [7], will be important in Section 5.

Using the delayed (co-)substitution notation, we can rewrite the cut-elimination rules of $\overline{\lambda}\mu\tilde\mu$ as follows:

$$\begin{array}{crcl}
(\beta_\supset) & \langle\lambda x.t|u :: e\rangle & \to & \langle u/x\rangle\langle t|e\rangle\\
(\sigma_{\tilde\mu}) & \langle t/x\rangle c & \to & [t/x]c\\
(\sigma_\mu) & \langle e/a\rangle c & \to & [e/a]c
\end{array}$$

This notation emphasizes that $\beta_\supset$, $\sigma_{\tilde\mu}$, and σ_μ are about generation ($\beta_\supset$) and execution of explicit (co-)substitution ($\sigma_{\tilde\mu}$, σ_μ). The execution itself is in one go, by calling meta-operations.[5] This is a half baked computational interpretation, that, in particular, says nothing about co-terms.

On the other hand, Curien and Herbelin [7] give intuitions about the expressions of $\overline{\lambda}\mu\tilde\mu$ in terms of "contexts" and "hole filling" derived from some (never specified) variant of the $\lambda\mu$-calculus, that form the basis of a totally different interpretation of $\overline{\lambda}\mu\tilde\mu$. More precisely:

(i) Co-terms e correspond to contexts $\mathcal{E}$, and the left type A in $\Gamma|e : A \vdash \Delta$ is the type of the hole of $\mathcal{E}$.

[4]We reuse in the context of $\overline{\lambda}\mu\tilde\mu$ notation traditionally used in other system, because: (i) it is handy to introduce abbreviations for complex expressions in $\overline{\lambda}\mu\tilde\mu$; (ii) it is unbearable to invent new notation for all these notions, or tag everywhere; (iii) it is mnemonic (and even illuminating) to re-use notation if the abstract concept is the same (albeit implemented in a different system). See Subsection 2.1 for more on notation, abbreviations and overloading.

[5]This view is our justification for using σ in the name of the reduction rules.

(ii) a corresponds to $a([\_])$.

(iii) $\tilde{\mu}x.c$ corresponds to $\mathsf{let}\,[\_] = x\,\mathsf{in}\,S$, if c corresponds to S.

(iv) $u :: e$ corresponds to $\mathcal{E}[[\_]N]$, if N (resp. $\mathcal{E}$) corresponds to u (resp. e)

(v) Cuts $\langle t|e\rangle$ correspond to filling $\mathcal{E}[M]$, if M (resp. $\mathcal{E}$) corresponds to t (resp. e)

The ideas (i), (iv) and (v) are explicit in [7]. The ideas (ii) and (iii) are implicit in the same paper, and explicit in [19].

3. The natural deduction system $\underline{\lambda}\mu\mathsf{let}$

In this section we introduce $\underline{\lambda}\mu\mathsf{let}$. First we define the logical/typing system, later the reduction/normalisation rules. As we present $\underline{\lambda}\mu\mathsf{let}$, we compare informally with Parigot's $\lambda\mu$. The precise connection with $\lambda\mu$ will be made in Section 5.

3.1. Definition of the system

Primitive syntax. Expressions of $\underline{\lambda}\mu\mathsf{let}$ are defined by the following grammar:

$$\begin{array}{lrcl} \text{(Terms)} & M,N,P & ::= & x \mid \lambda x.M \mid \mu a.S \\ \text{(Hole Expressions)} & H & ::= & \mathsf{h}(M) \mid HN \\ \text{(Statements)} & S & ::= & a(H) \mid \mathsf{let}\,x = H\,\mathsf{in}\,S \end{array}$$

Terms are either variables, λ-abstractions $\lambda x.M$, or μ-abstractions $\mu a.S$ whose body is a statement S. *Statements* are either *named expressions* of the form $a(H)$, or *let-expressions* $\mathsf{let}\,x = H\,\mathsf{in}\,S$. Hole expressions H are either *coercions*[6] $\mathsf{h}(M)$, or applications HN. A value V is a term of the form x or $\lambda x.M$.

The syntactic organization is subtle. Neither applications, nor let-expressions are terms. The same applies to the expression in the function position of applications, and to the two components of a let-expression. Hole expressions are indeed expressions that are filled in the hole of contexts (see below); but the expressions with holes, as usual, are derived syntax, whereas the expressions that go into holes are primitive.

In $\lambda\mu$ there are neither hole expressions, nor let-expressions. Applications are terms and statements are just named terms $a(M)$.

In $\underline{\lambda}\mu\mathsf{let}$, every statement has one of two forms

$$a(\mathsf{h}(M)N_1\cdots N_m)$$
$$\mathsf{let}\,x = (\mathsf{h}(M)N_1\cdots N_m)\,\mathsf{in}\,S$$

with $m \geq 0$. So, not only $\mathsf{h}(M)$ means M coerced to a hole expression, but it also signals the *head* term of a statement.

Primitive typing. The typing system of $\underline{\lambda}\mu\mathsf{let}$, given in Fig. 4, derives three kinds of sequents, one for each syntactic class:

$$\Gamma \vdash M : A|\Delta \qquad \Gamma \rhd H : A|\Delta \qquad S : (\Gamma \vdash \Delta)\ .$$

Figure 4: Typing rules for $\underline{\lambda}\mu\mathsf{let}$

$$\frac{}{\Gamma, x : A \vdash x : A|\Delta}\ \mathit{Assumption}$$

$$\frac{\Gamma \rhd H : A \supset B|\Delta \quad \Gamma \vdash N : A|\Delta}{\Gamma \rhd HN : B|\Delta}\ \mathit{Elim} \qquad \frac{\Gamma, x : A \vdash M : B|\Delta}{\Gamma \vdash \lambda x.M : A \supset B|\Delta}\ \mathit{Intro}$$

$$\frac{\Gamma \rhd H : A|\Delta, a : A}{a(H) : (\Gamma \vdash \Delta, a : A)}\ \mathit{Pass} \qquad \frac{S : (\Gamma \vdash \Delta, a : A)}{\Gamma \vdash \mu a.S : A|\Delta}\ \mathit{Act}$$

$$\frac{\Gamma \vdash M : A|\Delta}{\Gamma \rhd \mathsf{h}(M) : A|\Delta}\ \mathit{Coercion}$$

$$\frac{\Gamma \rhd H : A|\Delta \quad S : (\Gamma, x : A \vdash \Delta)}{\mathsf{let}\, x = H \,\mathsf{in}\, S : (\Gamma \vdash \Delta)}\ \mathit{Subst}$$

The first and third kinds (*term sequents* and *statement sequents*, resp.) are familiar from $\lambda\mu$. If we disregard the distinction between the first two kinds of sequents, then the first five typing rules in Fig. 4 are exactly those of $\lambda\mu$, and the coercion rule is a trivial repetition rule. So, up to the final substitution rule, we have a refinement of the typing system of $\lambda\mu$, a classical natural deduction system with three kinds of sequents instead of two, and containing an extra-rule for coercing between two different kinds of sequents.

The existence of an extra form of sequents $\Gamma \rhd H : A|\Delta$, rather than an obscure complication, is an advantage of the system, because of the role in the subformula property, to be proved below: it allows a proof by induction on normal forms, very much in the style of sequent calculus, and typically impossible for simpler formulations of natural deduction[7].

The final rule in Fig. 4 is a primitive *substitution inference rule*, which is the typing rule for let-expressions. The inference rule is standard, apart from the fact that sequents have to be chosen of the appropriate kind. One might argue that a substitution rule is a cut rule. Does this last rule disturb the natural deduction character of the system? The answer is "no" but can only be given later. First, we need to see how the isomorphism between $\overline{\lambda}\mu\tilde{\mu}$ and $\underline{\lambda}\mu\mathsf{let}$ links cuts and substitutions. Only then we compare the two inference principles (in Subsection 4.2).

A *typable* term is a term M such that $\Gamma \vdash M : A|\Delta$ is derivable, for some Γ, Δ, A. Similarly for hole expressions H and statements S.

Derived syntax. We define:

[6]At the level of expressions, a "coercion" forces an expression from one syntactic class to another. We follow [6] in this use of the terminology "coercion". There will be a corresponding coercion typing rule.

[7]Recall that Prawitz [28] first proves the structure of "branches" in normal deduction, and then proves the subformula property by induction on the "order" of branches.

$$\begin{array}{rcll} MN & := & \mu a.a(\mathsf{h}(M)N) & \text{(ordinary application)} \\ a(M) & := & a(\mathsf{h}(M)) & \text{(named term)} \\ \langle N/x\rangle S & := & \mathsf{let}\, x = \mathsf{h}(N)\, \mathsf{in}\, S & \text{(delayed substitution)} \end{array}$$

The first two definitions allow an immediate and simple translation of $\lambda\mu$ into $\underline{\lambda}\mu\mathsf{let}$ (in the first definition, of course, a is fresh). A different translation of $\lambda\mu$, based on a different form of deriving ordinary application, is possible. Both will be studied in Subsection 5.4. In the third definition, we propose the concept of delayed substitution as a particular case of let-expressions. Delayed substitution will be used in explaining the reduction rules of $\underline{\lambda}\mu\mathsf{let}$ soon in this section, and in characterizing the CBN fragments of $\underline{\lambda}\mu\mathsf{let}$ (Section 5).

Another derived syntactical concept of $\underline{\lambda}\mu\mathsf{let}$, crucial for the definition of reduction rules and for the comparison with $\overline{\lambda}\mu\tilde{\mu}$, is that of *context*. A context is an expression of the two possible forms ($m \geq 0$):

$$a([\_]N_1 \cdots N_m) \qquad \mathsf{let}\, x = ([\_]N_1 \cdots N_m)\, \mathsf{in}\, S \tag{2}$$

So, a context is a statement with a "hole" $[\_]$ in a position where a hole expression H is expected. Let $\mathcal{E}$ range over contexts, and $\mathcal{E}[H]$ denote the statement obtained by filling the hole of $\mathcal{E}$ with H. Notice that, if we fill the hole of $\mathcal{E}$ with $[\_]N$ we obtain another context[8].

In addition to meta-substitution for variables, there are three operations of meta-substitution for co-variables $[\mathcal{E}/a]M$, $[\mathcal{E}/a]H$, and $[\mathcal{E}/a]S$, defined by a simultaneous recursion, all of whose clauses are homomorphic, but the crucial one:

$$[\mathcal{E}/a](a(H)) = \mathcal{E}[H'] \qquad \text{where } H' = [\mathcal{E}/a]H.$$

For instance:

(i) $[b([\_]N)/a](a(\mathsf{h}(M))) = b(\mathsf{h}(M')N)$, with $M' = [b([\_]N)/a]M$; so, $[b([\_]N)/a]\_$ is a form of "structural substitution" as found in $\lambda\mu$.

(ii) $[\mathcal{E}[[\_]N]/a](a(\mathsf{h}(M))) = \mathcal{E}[\mathsf{h}(M')N]$, with $M' = [\mathcal{E}[[\_]N]/a]M$; this is a generalization of the previous case.

(iii) $[b([\_])/a](a(H)) = b(H')$, with $H' = [b([\_])/a]H$; so $[b([\_])/a]\_$ is a renaming operation, also found in $\lambda\mu$.

Reduction rules. Some of the reduction rules of $\underline{\lambda}\mu\mathsf{let}$ will act on the head of statements. We use contexts as a device for bringing to surface such heads, which normally are buried under a sequence of arguments. For instance, if S is $\mathsf{let}\, x = \mathsf{h}(M)N_1 \cdots N_m\, \mathsf{in}\, S'$, then $S = \mathcal{E}[\mathsf{h}(M)]$, where $\mathcal{E} = \mathsf{let}\, x = [\_]N_1 \cdots N_m\, \mathsf{in}\, S'$.

The reduction rules of $\underline{\lambda}\mu\mathsf{let}$ are given in Figure 5.[9] By *normalisation* we mean $\beta_{\supset}\sigma_{\mathsf{let}}\sigma_{\mu}$-reduction. In addition to the normalisation rules, there are three η-like rules. The discussion of rule $\eta_{\supset}$ is entirely postponed to Subsection 6.5.

[8]In fact, what we have just said of $[\_]N$ can be said of $[\_]\vec{N}$ (an H with a hole in its left end), but we are not going to make use of this more general possibility.

[9]In spite of the use of contexts, there should be no misunderstanding: these reduction rules, as well as their compatible closure etc., are relations on the primitive syntax of $\underline{\lambda}\mu\mathsf{let}$, not on some extended

Figure 5: Reduction rules of $\underline{\lambda}\mu\mathsf{let}$

$$\begin{array}{rrcll}
(\beta_\supset) & \mathcal{E}[\mathsf{h}(\lambda x.M)N] & \to & \langle N/x\rangle\mathcal{E}[\mathsf{h}(M)] & \\
(\sigma_{\mathsf{let}}) & \langle N/x\rangle S & \to & [N/x]S & \\
(\sigma_\mu) & \mathcal{E}[\mathsf{h}(\mu a.S)] & \to & [\mathcal{E}/a]S & \\
\\
(\eta_\supset) & \lambda x.\mu a.a(Hx) & \to & \mu a.a(H) & (x,a\notin H)\\
(\eta_{\mathsf{let}}) & \mathsf{let}\,x=H\,\mathsf{in}\,\mathcal{E}[\mathsf{h}(x)] & \to & \mathcal{E}[H] & (x\notin\mathcal{E})\\
(\eta_\mu) & \mu a.a(\mathsf{h}(M)) & \to & M & (a\notin M)
\end{array}$$

Rule $\beta_\supset$ generates a delayed substitution, that jumps out of the statement that constituted the redex. Such delayed substitution is executed by a separate rule (σ_{let}). This reading is obscured if we unfold the definition of delayed substitution and write the "low level"[10] definition of the rules in terms of the primitive syntax:

$$\begin{array}{rrcl}
(\beta_\supset) & \mathcal{E}[\mathsf{h}(\lambda x.M)N] & \to & \mathsf{let}\,x=\mathsf{h}(N)\,\mathsf{in}\,\mathcal{E}[\mathsf{h}(M)]\\
(\sigma_{\mathsf{let}}) & \mathsf{let}\,x=\mathsf{h}(N)\,\mathsf{in}\,S & \to & [N/x]S
\end{array}$$

This version makes it clear that $\beta_\supset$-reduction generates a let-expression, as is common with CBV λ-calculi, although does not emphasize that the let-expression has a particular form, and that this particular form has a meaning. We will see in Section 5, and has been said before, that such particular form of let-expressions characterizes CBN fragments [11].

The rule σ_{let} can be given yet another presentation:

$$(\sigma_{\mathsf{let}})\quad \mathcal{E}[\mathsf{h}(N)] \quad\to\quad [N/x]S,\qquad \text{if } \mathcal{E}=\mathsf{let}\,x=[\_]\,\mathsf{in}\,S \tag{3}$$

This has the advantage of making uniform the style of the three normalisation rules, as rules that take a statement, inspect its head, and reduce accordingly[12]. This presentation also makes it plain that there is a $\sigma_{\mathsf{let}}/\sigma_\mu$ critical pair, as shown in Figure 6. In Section 5 it will become clear that this critical pair corresponds to the $\sigma_{\tilde{\mu}}/\sigma_\mu$ critical pair of $\overline{\lambda}\mu\tilde{\mu}$. For this reason, we also refer to a CBN/CBV dilemma in $\underline{\lambda}\mu\mathsf{let}$.

---

syntax. For instance, $\beta_\supset$ could be defined as the union of two rules:

$$\begin{array}{rcl}
a(\mathsf{h}(\lambda x.M)N\vec{N}) & \to & \langle N/x\rangle(a(\mathsf{h}(M)\vec{N}))\\
\mathsf{let}\,x=\mathsf{h}(\lambda x.M)N\vec{N}\,\mathsf{in}\,S & \to & \langle N/x\rangle(\mathsf{let}\,x=\mathsf{h}(M)\vec{N}\,\mathsf{in}\,S)
\end{array}$$

There are many disadvantages of this style of definition: it is less compact; the use of vectors $\vec{N}$ makes it informal; it does not convey the manipulation of contexts that makes $\underline{\lambda}\mu\mathsf{let}$ a control calculus.

[10] One example of "low-level-ness" in the following version of σ_{let} is that we see that the term N in the hole expression $\mathsf{h}(N)$ has to be stripped of the coercion before it is substituted for the term variable x.

[11] So, the $\beta_\supset$-contractum shows some CBN/CBV ambiguity. Actually, if in the $\beta_\supset$-redex the body of the λ-abstraction is not a value, and $\mathcal{E}=\mathsf{let}\,x=[\_]\,\mathsf{in}\,S$, then the $\beta_\supset$-contractum generates immediately a $\sigma_{\mathsf{let}}/\sigma_\mu$ critical pair. This CBN/CBV ambiguity does not surprise, since $\underline{\lambda}\mu\mathsf{let}$, as $\overline{\lambda}\mu\tilde{\mu}$, will "superimpose" [19] CBN and CBV.

[12] This is the *modus operandi* of the reduction *rules*, that is, reduction at root position. Of course, these rules generate reduction relations by compatible closure, and the statement inspected can be at arbitrary position in a expression.

Figure 6: CBN/CBV dilemma in $\underline{\lambda}\mu\mathsf{let}$

$$\begin{tikzcd}
{[\mathsf{let}\, x = [\_]\,\mathsf{in}\, S'/a]S} & \mathsf{let}\, x = \mathsf{h}(\mu a.S)\,\mathsf{in}\, S' \arrow[l, "\sigma_\mu"'] \arrow[r, "\sigma_{\mathsf{let}}"'] & {[\mu a.S/x]S'}
\end{tikzcd}$$

Rule σ_μ plays in $\underline{\lambda}\mu\mathsf{let}$ a role similar to the role played by rules μ and ρ in $\lambda\mu$, being the union of two rules:

$$\begin{array}{rcll}
b((\mathsf{h}(\mu a.S)\vec{N})) & \to & [b([\_]\vec{N})/a]S & (4) \\
\mathsf{let}\, x = \mathsf{h}(\mu a.S)\vec{N}\,\mathsf{in}\, S' & \to & [\mathsf{let}\, x = [\_]\vec{N}\,\mathsf{in}\, S'/a]S\ , & (5)
\end{array}$$

with $\vec{N}$ of length $m \geq 0$. In (4), if $m = 0$ then we have a version of the "renaming" rule ρ; allowing $m \geq 0$ gives a more general variant already considered in [2]. In (5), if $m > 0$ then a subexpression of the form $\mathsf{h}(\mu a.S)N$ exists; but, in contrast to rule μ of $\lambda\mu$, the whole statement of which $\mathsf{h}(\mu a.S)$ is the head is transformed in a single σ_μ-step.

Rule η_μ is similar to the rule with same name in $\lambda\mu$. Rule η_{let} has no counterpart in $\lambda\mu$ because the latter has no let-expressions.

Finally a technical remark. In general, a statement S can be decomposed as $\mathcal{E}[H]$ for many choices of the pair $\mathcal{E}$, H; however, in the redexes for rules $\beta_\supset$, σ_μ, η_{let}, and σ_{let} in the version (3), $\mathcal{E}$ is uniquely determined.

3.2. Properties of the system

Strong normalisation of typable expressions and subject reduction will be a consequence of isomorphism with $\overline{\lambda}\mu\tilde{\mu}$, to be proved in the next section. Another consequence of isomorphism is that the $\sigma_{\mathsf{let}}/\sigma_\mu$ critical pair breaks confluence of $\underline{\lambda}\mu\mathsf{let}$, in the same way as the $\sigma_{\tilde{\mu}}/\sigma_\mu$ critical pair breaks confluence of $\overline{\lambda}\mu\tilde{\mu}$. Later (see Section 5) we will discuss fragments of $\underline{\lambda}\mu\mathsf{let}$ that are isomorphic to confluent fragments of $\overline{\lambda}\mu\tilde{\mu}$, and therefore confluent themselves.

The $\beta_\supset\sigma_\mu\sigma_{\mathsf{let}}$-normal forms are given by:

$$\begin{array}{rcl}
M_{nf}, N_{nf} & ::= & x \mid \lambda x.M_{nf} \mid \mu a.S_{nf} \\
H_{nf} & ::= & \mathsf{h}(x) \mid H_{nf}N_{nf} \\
S_{nf} & ::= & a(\mathsf{h}(\lambda x.M_{nf})) \\
 & \mid & a(H_{nf}) \mid \mathsf{let}\, x = H_{nf}N_{nf}\,\mathsf{in}\, S_{nf}
\end{array}$$

At the level of derivations, the *normality criterion* is:

- The left premiss of every substitution is the conclusion of an elimination;
- The premiss of a coercion is never the conclusion of an activation; moreover if a coercion is the main premiss of an elimination, then its premiss is an assumption.

We say that A is a subformula of Γ (resp. Δ) if there is some declaration $x : B$ (resp. $a : B$) in Γ (resp. Δ) such that A is a subformula of B.

Theorem 1 (Subformula property).

1. *In a derivation of* $\Gamma \vdash M_{nf} : A|\Delta$*, all formulas are subformulas of* Γ*,* A*, or* Δ*.*
2. *In a derivation of* $S_{nf} : (\Gamma \vdash \Delta)$*, all formulas are subformulas of* Γ *or* Δ*.*
3. *In a derivation of* $\Gamma \rhd H_{nf} : A|\Delta$*, (a) all formulas are subformulas of* Γ *or* Δ*; and (b)* A *is a subformula of* Γ*.*

Proof: Before we embark on it, we do an analysis of the typing rules in Fig. 4. Let $\phi_1(M_{nf})$, $\phi_2(S_{nf})$, and $\phi_3(H_{nf})$ be the three statements in the theorem we are proving. Notice we can ask $\phi_1(M)$ of any term (not necessarily normal), similarly for $\phi_2(S)$ and $\phi_3(H)$. But we cannot ask $\phi_2(M)$, because ϕ_2 applies to statements and is not "appropriate" to terms.

Let us say that a typing rule of $\underline{\lambda}\mu\mathsf{let}$ *preserves the subformula property* if the expression in the conclusion of the rule satisfies the appropriate ϕ_i whenever the expressions in the premisses satisfy the appropriate ϕ_i's. Notice that the expressions in the premisses and conclusions of the rules are not necessarily normal. For instance, the rule *Act* preserves the subformula property if $\phi_2(S)$ implies $\phi_1(\mu a.S)$. A simple rule inspection shows that all rules but *Coercion* preserve the subformula property. There is nothing to prove for *Assumption*, *Intro*, *Act*, and *Pass*, as all formulas in the premis(ses) stay unchanged, or are replaced by superformulas, in the conclusion. The proviso 3.(b) is enough for *Elim* and *Subst*. For instance, although the substitution formula A disappears in the conclusion of *Subst*, it is in Γ by the proviso. The rule *Coercion* is different, because $\phi_1(M)$ guarantees part (a) but not part (b) of $\phi_3(\mathsf{h}(M))$.

Now the proof of the theorem: it is by simultaneous induction on M_{nf}, S_{nf}, and H_{nf}. According to the inductive definition of normal forms, there are 8 cases. All cases in the proof which do not mention $\mathsf{h}(.)$ are done, by IH and the fact that all rules different from *Coercion* preserve the subformula property. The two cases remaining are $H_{nf} = \mathsf{h}(x)$ and $S_{nf} = a(\mathsf{h}(\lambda x.M_{nf}))$. For these we check the respective claims by analyzing the respective derivations. This is trivial for H_{nf} (yes, in the case $\mathsf{h}(x)$ part (b) can be guaranteed); for S_{nf} the IH for M_{nf} suffices. □

3.3. Discussion

Maybe the reader finds some aspects of the design of $\underline{\lambda}\mu\mathsf{let}$ unnecessarily complex or even unfortunate. But recall that there is no real freedom to design the system as we please or think is right; the design of the system is dictated by the axiom that $\underline{\lambda}\mu\mathsf{let}$ is to be the isomorphic reflection of $\overline{\lambda}\mu\tilde{\mu}$ in natural deduction. However, a closer inspection reveals an intrinsic justification of many aspects of $\underline{\lambda}\mu\mathsf{let}$.

Consider the reduction rules of $\underline{\lambda}\mu\mathsf{let}$. We might regret, at first sight, their complex formulation, with explicit manipulation of contexts $\mathcal{E}$. A more careful consideration shows some of their virtues. First, recall that $\underline{\lambda}\mu\mathsf{let}$, being isomorphic to $\overline{\lambda}\mu\tilde{\mu}$ has to encompass both CBN and CBV computation, and that CBV computation is "de facto complex to describe" [20] - as is seen from the axiomatizations in [32, 20]. Compared to these, the set of reduction rules of $\underline{\lambda}\mu\mathsf{let}$ is relatively mild, as we do not see big sets of permutative rules. Indeed, the set of reduction rules of $\underline{\lambda}\mu\mathsf{let}$ has a pattern that is very close to that of $\lambda\mu$, with σ_μ even abstracting the two rules μ and ρ, and a similar η_μ-rule. The split of β into a rule for generating and another for executing substitution is a familiar design. Second, still on explicit manipulation of contexts $\mathcal{E}$: isn't $\underline{\lambda}\mu\mathsf{let}$ supposed to be a control calculus? The rule σ_μ is particularly important: it expresses the features of μ-abstraction as a control operator. Once σ_μ of $\underline{\lambda}\mu\mathsf{let}$ is proved isomorphic to σ_μ of $\overline{\lambda}\mu\tilde{\mu}$, we will finally understand in what sense the μ-abstraction of $\overline{\lambda}\mu\tilde{\mu}$ is a control operator.

Consider now the organization of $\underline{\lambda}\mu\mathsf{let}$'s expressions into three syntactic classes: terms, hole-expressions, and statements. The same number of classes is seen in $\overline{\lambda}\mu\tilde{\mu}$ (terms, co-terms, and commands), but in $\overline{\lambda}\mu\tilde{\mu}$ the perfect symmetry imposes the design. The isomorphic design of $\underline{\lambda}\mu\mathsf{let}$ delivers a syntax with the same number of classes, but it seemingly lost the term/co-term symmetry. However, if $\underline{\lambda}\mu\mathsf{let}$ is isomorphic to $\overline{\lambda}\mu\tilde{\mu}$, the symmetry has to be there: the point is that the symmetry is not necessarily observed in the primitive syntax. In fact, once the isomorphism $\overline{\lambda}\mu\tilde{\mu} \cong \underline{\lambda}\mu\mathsf{let}$ is sufficiently analyzed in Section 4, the term/co-term symmetry of $\overline{\lambda}\mu\tilde{\mu}$, which *prima facie* is just a formal symmetry, and whose informal semantics [7] is a symmetry between terms and "contexts", is recovered in $\underline{\lambda}\mu\mathsf{let}$ as the symmetry between terms and contexts.

Moreover, the design of $\underline{\lambda}\mu\mathsf{let}$ with its three syntactic classes can be justified without reference to $\overline{\lambda}\mu\tilde{\mu}$, as an internal necessity of the development of natural deduction as a "coercion calculus". This terminology is due to [6], where one such calculus is introduced as an auxiliary tool in the study of the meta-theory of a fragment of sequent calculus[13]. Essentially, the coercion calculus of [6] changes the definition of the ordinary λ-calculus

$$M, N \ ::= \ x \,|\, \lambda x.M \,|\, MN \ ,$$

to

$$M, N \ ::= \ x \,|\, \lambda x.M \,|\, \{H\} \qquad H \ ::= \ \mathsf{h}(M) \,|\, HN \ ,$$

where the separation into several syntactic classes starts, and where MN is developed into a *backward coercion* $\{H\}$, that forces an H back to the class of terms. In [10], the author recognized the idea of coercion calculus as a principle for the design of natural deduction systems isomorphic to successively larger fragments of the sequent calculus. In order to reflect full intuitionistic sequent calculus, the next step [12] was to develop $\{H\}$ into a construction $\mathsf{let}\, x = H \,\mathsf{in}\, P$[14], yielding:

$$M, N, P \ ::= \ x \,|\, \lambda x.M \,|\, \mathsf{let}\, x = H \,\mathsf{in}\, P \qquad H \ ::= \ \mathsf{h}(M) \,|\, HN \ .$$

$\underline{\lambda}\mu\mathsf{let}$ represents a further elaboration into $\mu a.S$ of the same construction

$$M, N, P \ ::= \ x \,|\, \lambda x.M \,|\, \mu a.S \qquad S \ ::= \ a(H) \,|\, \mathsf{let}\, x = H \,\mathsf{in}\, S \qquad H \ ::= \ \mathsf{h}(M) \,|\, HN \ ,$$

capable of capturing full classical sequent calculus, but requiring the introduction of a third syntactic class. The progression

$$\begin{array}{rcl} MN & := & \{\mathsf{h}(M)N\} \\ \{H\} & := & \mathsf{let}\, x = H \,\mathsf{in}\, x \\ \mathsf{let}\, x = H \,\mathsf{in}\, P & := & \mu a.\mathsf{let}\, x = H \,\mathsf{in}\, a(\mathsf{h}(P)) \end{array}$$

[13]The sequent calculus of [6] is called the "spine calculus". The part of it concerned with intuitionistic implication corresponds to the fragment of sequent calculus studied in [18].

[14]In [12], $\mathsf{let}\, x = H \,\mathsf{in}\, P$ is written $\{H/x\}P$.

determines the inclusions among the four successive natural deduction systems just mentioned; it also determines the successive *split* of the original concept of application.

Consider, finally, the proof of the subformula property. In Gentzen's LK [15], a proof of the subformula property by rule inspection works: in any inference rule different from the cut rule, all formulas occurring in the premisses are repeated or replaced by superformulas in the conclusions. As we have seen, in $\underline{\lambda}\mu\mathsf{let}$ inspection of inference rules falls short. Likewise, inspection of inference rules is also insufficient for establishing the subformula property for $\overline{\lambda}\mu\tilde{\mu}$, because some instances of Cut are not eliminable. Nevertheless, in $\underline{\lambda}\mu\mathsf{let}$ a proof by induction on normal forms is possible and straightforward; in addition, like in LK and $\overline{\lambda}\mu\tilde{\mu}$ the problem with the subformula property in localized in a single inference rule: the cut rule in the case of sequent calculus, the rule $Coercion$ in the case of $\underline{\lambda}\mu\mathsf{let}$.

4. Isomorphism

In this section mappings $\Theta : \overline{\lambda}\mu\tilde{\mu} \to \underline{\lambda}\mu\mathsf{let}$ and $\Psi : \underline{\lambda}\mu\mathsf{let} \to \overline{\lambda}\mu\tilde{\mu}$ are defined and analyzed. We show why Θ is a semantics of $\overline{\lambda}\mu\tilde{\mu}$, and explain the proof-theory of let-expressions, delayed substitutions and named expressions. Next we establish $\overline{\lambda}\mu\tilde{\mu} \cong \underline{\lambda}\mu\mathsf{let}$. As a corollary, strong normalisation and subject reduction for $\underline{\lambda}\mu\mathsf{let}$ follow.

4.1. Mappings Θ and Ψ.

We start with an informal explanation. Let $\Theta t = M$, $\Theta(u_i) = N_i$ and $\Theta c = S$[15]. The idea behind $\Theta : \overline{\lambda}\mu\tilde{\mu} \longrightarrow \underline{\lambda}\mu\mathsf{let}$ is to map cuts as follows:

$$\begin{aligned} \langle t|u_1 :: \cdots :: u_m :: \tilde{\mu}x.c\rangle &\mapsto \mathsf{let}\, x = \mathsf{h}(M)N_1 \cdots N_m \,\mathsf{in}\, S & (6)\\ \langle t|u_1 :: \cdots :: u_m :: a\rangle &\mapsto a((\mathsf{h}(M)N_1 \cdots N_m)) & (7)\end{aligned}$$

The idea behind $\Psi : \underline{\lambda}\mu\mathsf{let} \longrightarrow \overline{\lambda}\mu\tilde{\mu}$ is the translation of statements obtained by reverting these mappings, with $\Psi M = t$, $\Psi(N_i) = u_i$ and $\Psi S = c$. Observe that, in (6) and (7), each occurrence of left introduction $u_i :: e_i$ is replaced by an occurrence application $H_i N_i$. Conversely for Ψ. Moreover, such replacement inverts the right associativity of the l.h.s. expressions of (6) and (7) - with the head term t at the surface - to the left associativity of the r.h.s. expressions of (6) and (7) - with the head term M buried under m applications.

Both Θ and Ψ consist of three mappings defined by simultaneous recursion in Fig. 7. The morphism $\Theta : \overline{\lambda}\mu\tilde{\mu} \longrightarrow \underline{\lambda}\mu\mathsf{let}$ consists of three mappings

- $\Theta : \overline{\lambda}\mu\tilde{\mu} - Terms \longrightarrow \underline{\lambda}\mu\mathsf{let} - Terms$,
- $\Theta : \overline{\lambda}\mu\tilde{\mu} - Commands \longrightarrow \underline{\lambda}\mu\mathsf{let} - Statements$,
- $\Theta : \underline{\lambda}\mu\mathsf{let} - HoleExpressions \times \overline{\lambda}\mu\tilde{\mu} - Contexts \longrightarrow \underline{\lambda}\mu\mathsf{let} - Statements$

The morphism $\Psi : \underline{\lambda}\mu\mathsf{let} \longrightarrow \overline{\lambda}\mu\tilde{\mu}$ consists of three mappings

[15]As usual in mathematics, we omit parentheses in function application when no confusion arises.

Figure 7: Mappings $\Theta : \overline{\lambda}\mu\tilde{\mu} \rightarrow \underline{\lambda}\mu\mathsf{let}$ and $\Psi : \underline{\lambda}\mu\mathsf{let} \rightarrow \overline{\lambda}\mu\tilde{\mu}$

Θx	$=$	x
$\Theta(\lambda x.t)$	$=$	$\lambda x.\Theta t$
$\Theta(\mu a.c)$	$=$	$\mu a.\Theta c$
$\Theta\langle t\vert e\rangle$	$=$	$\Theta(\mathsf{h}(\Theta t), e)$
$\Theta(H, a)$	$=$	$a(H)$
$\Theta(H, \tilde{\mu}x.c)$	$=$	$\mathsf{let}\, x = H \,\mathsf{in}\, \Theta c$
$\Theta(H, u :: e)$	$=$	$\Theta(H\Theta u, e)$

Ψx	$=$	x
$\Psi(\lambda x.M)$	$=$	$\lambda x.\Psi M$
$\Psi(\mu a.S)$	$=$	$\mu a.\Psi S$
$\Psi(a(H))$	$=$	$\Psi(H, a)$
$\Psi(\mathsf{let}\, x = H \,\mathsf{in}\, S)$	$=$	$\Psi(H, \tilde{\mu}x.\Psi S)$
$\Psi(\mathsf{h}(M), e)$	$=$	$\langle \Psi M\vert e\rangle$
$\Psi(HN, e)$	$=$	$\Psi(H, \Psi N :: e)$

- $\Psi : \underline{\lambda}\mu\mathsf{let} - Terms \longrightarrow \overline{\lambda}\mu\tilde{\mu} - Terms$,
- $\Psi : \underline{\lambda}\mu\mathsf{let} - Statements \longrightarrow \overline{\lambda}\mu\tilde{\mu} - Commands$,
- $\Psi : \underline{\lambda}\mu\mathsf{let} - HoleExpressions \times \overline{\lambda}\mu\tilde{\mu} - Contexts \longrightarrow \overline{\lambda}\mu\tilde{\mu} - Commands$

Proposition 1 (Soundness). *The typing rules of Figure 8 are admissible.*[16]

Proof: The left rules are proved by simultaneous induction on the t, c, and e, using inversion of the typing rules of $\overline{\lambda}\mu\tilde{\mu}$, which is obvious, since the typing system of $\overline{\lambda}\mu\tilde{\mu}$ is syntax-directed. The right rules are proved by simultaneous induction on M, S, and H, using inversion of the typing rules of $\underline{\lambda}\mu\mathsf{let}$, which again is obvious, as the typing system of $\underline{\lambda}\mu\mathsf{let}$ is syntax-directed. We just show the first induction.

The cases $t = x$, $t = \lambda x.t'$, and $t = \mu a.c$ are straightforward.

Case $c = \langle t|e\rangle$. Suppose $\langle t|e\rangle : (\Gamma \vdash \Delta)$. We get $\Gamma \vdash t : A|\Delta$ and $\Gamma|e : A \vdash \Delta$, for some A. The IH for t gives $\Gamma \vdash \Theta t : A|\Delta$, from which we get $\Gamma \rhd \mathsf{h}(\Theta t) : A|\Delta$, by *Coercion*. This last sequent, together with IH for e yields $\Theta(\mathsf{h}(\Theta t), e) : (\Gamma \vdash \Delta)$. We are done, since $\Theta(\mathsf{h}(\Theta t), e) = \Theta(\langle t|e\rangle)$.

Case $e = a$. Suppose $\Gamma|a : A \vdash a : A, \Delta$ and $\Gamma \rhd H : A|a : A, \Delta$. From the last sequent we get $a(H) : (\Gamma \vdash a : A, \Delta)$ by *Pass*. We are done, since $a(H) = \Theta(H, a)$.

Case $e = \tilde{\mu}x.c$. Suppose $\Gamma|\tilde{\mu}x.c : A \vdash \Delta$ and $\Gamma \rhd H : A|\Delta$. From the first sequent we get $c : (\Gamma, x : A \vdash \Delta)$. By IH for c we get $\Theta c : (\Gamma, x : A \vdash \Delta)$. From this and the typing of H we get, by *Subst*, $\mathsf{let}\, x = H \,\mathsf{in}\, \Theta c : (\Gamma \vdash \Delta)$. We are done, since $\mathsf{let}\, x = H \,\mathsf{in}\, \Theta c = \Theta(H, \tilde{\mu}x.c)$.

Case $e = u :: e'$. Suppose $\Gamma|u :: e' : A \supset B \vdash \Delta$ and $\Gamma \rhd H : A \supset B|\Delta$. From the first sequent we get $\Gamma \vdash u : A|\Delta$ and $\Gamma|e' : B \vdash \Delta$. From the IH for u we get $\Gamma \vdash \Theta u : A|\Delta$. From this and the typing of H we get, by *Elim*, $\Gamma \rhd H(\Theta u) : B|\Delta$. From this and the IH for e' we get $\Theta(H(\Theta u), e') : (\Gamma \vdash \Delta)$. We are done, since $\Theta(H(\Theta u), e') = \Theta(H, u :: e')$.
$\square$

As a consequence, Θ and Ψ preserve typability.

Proposition 2 (Bijections). Θ *and* Ψ *are inverse bijections at the level of terms and commands/statements. More precisely:*

[16]It is an abuse of language to call the implications in Figure 8 "admissible typing rules", since premisses and conclusions live in different system. Yet it is a harmless abuse that we are going to repeat.

Figure 8: Soundness of Θ and Ψ

$\dfrac{\Gamma \vdash t : A \mid \Delta}{\Gamma \vdash \Theta t : A \mid \Delta}$	$\dfrac{\Gamma \vdash M : A \mid \Delta}{\Gamma \vdash \Psi M : A \mid \Delta}$
$\dfrac{c : (\Gamma \vdash \Delta)}{\Theta c : (\Gamma \vdash \Delta)}$	$\dfrac{S : (\Gamma \vdash \Delta)}{\Psi S : (\Gamma \vdash \Delta)}$
$\dfrac{\Gamma \rhd H : A \mid \Delta \quad \Gamma \mid e : A \vdash \Delta}{\Theta(H, e) : (\Gamma \vdash \Delta)}$	$\dfrac{\Gamma \rhd H : A \mid \Delta \quad \Gamma \mid e : A \vdash \Delta}{\Psi(H, e) : (\Gamma \vdash \Delta)}$

1. *$\Theta\Psi M = M$ and $\Theta\Psi(H, e) = \Theta(H, e)$ (for all e) and $\Theta\Psi S = S$.*
2. *$\Psi\Theta t = t$ and $\Psi\Theta(H, e) = \Psi(H, e)$ (for all H) and $\Psi\Theta c = c$*

Proof: The first statement is by simultaneous induction on M, H and S. The second by simultaneous induction on t, e and c. We just show the first induction.

Cases $M = x$, $M = \lambda x.M'$ and $M = \mu a.S$ are straightforward.

Case $H = \mathsf{h}(M)$. $\Theta\Psi(\mathsf{h}(M), e) = \Theta(\langle \Psi M | e \rangle) = \Theta(\mathsf{h}(\Theta\Psi M), e) \overset{*}{=} \Theta(\mathsf{h}(M), e)$, where the marked equality is by IH for M.

Case $H = H'N$. $\Theta\Psi(H'N, e) = \Theta\Psi(H', \Psi N :: e) \overset{*}{=} \Theta(H', \Psi N :: e) = \Theta(H'N, e)$, where the marked equality is by IH for H'.

Case $S = a(H)$. $\Theta\Psi(a(H)) = \Theta\Psi(H, a) \overset{*}{=} \Theta(H, a) = aH$, where the marked equality is by IH for H.

Case $S = \mathsf{let}\, x = H \,\mathsf{in}\, S'$. $\Theta\Psi(\mathsf{let}\, x = H \,\mathsf{in}\, S') = \Theta\Psi(H, \tilde{\mu}x.\Psi S') \overset{*}{=} \Theta(H, \tilde{\mu}x.\Psi S') = \mathsf{let}\, x = H \,\mathsf{in}\, \Theta\Psi S' \overset{**}{=} \mathsf{let}\, x = H \,\mathsf{in}\, S'$, where the first marked equality is by IH for H and the second by IH for S'. □

4.2. Proof-theoretical foundation for let-expressions and named expressions.

Recall the typing systems of $\overline{\lambda}\mu\tilde{\mu}$ and $\underline{\lambda}\mu\mathsf{let}$ (Figs. 2 and 4). We explain the difference between cut in $\overline{\lambda}\mu\tilde{\mu}$ and the substitution inference rule in $\underline{\lambda}\mu\mathsf{let}$, thereby clarifying the proof-theoretical status of let-expressions, and completing the discussion of the typing system of $\underline{\lambda}\mu\mathsf{let}$ in Subsection 3.1. We also argue that delayed substitutions and Parigot's named terms are, in some sense, neutral w.r.t. the sequent calculus/natural deduction difference.

Recall the correspondences (6) and (7), and that a cut of the form of the l.h.s. of (6), resp. (7), is what we have called in Subsection 2.3 a $\tilde{\mu}$-cut, resp. a covar-cut. So, there is a bijective correspondence between $\tilde{\mu}$-cuts and let-expressions, and between covar-cuts and named expressions. Also, observe that

$$\langle t/x \rangle c \quad \mapsto \quad \langle M/x \rangle S \qquad (8)$$

$$a(t) \quad \mapsto \quad a(M) \qquad (9)$$

are particular cases of (6) and (7), respectively.

In contrast to the right premiss e of a $\tilde{\mu}$-cut $\langle t|e\rangle$, the right premiss S of the substitution inference $\mathsf{let}\, x = H \,\mathsf{in}\, S$ can never be the conclusion of left-introduction (similar remark already made by Negri and von Plato [23], pg. 172). But, on the other hand, the left premiss H is not limited to be morally a term $\mathsf{h}(M)$ (as is the left premiss t in $\langle t|e\rangle$), it can be a sequence of eliminations. So, $\tilde{\mu}$-cuts (in sequent calculus) are more general on the right premiss, while substitutions (in natural deduction) are more general on the left premiss. But there is a kind of common, neutral case, when $e = \tilde{\mu}x.c$ (so e is not a left introduction - the distinctive sequent calculus inference), and $H = \mathsf{h}(M)$ (so H is not an elimination - the distinctive natural deduction inference). Then the $\tilde{\mu}$-cut has the form $\langle t/x\rangle c = \langle t|\tilde{\mu}x.c\rangle$ and the substitution has the form $\langle M/x\rangle S = \mathsf{let}\, x = \mathsf{h}(M) \,\mathsf{in}\, S$, the form of a delayed substitution.

Similarly, a covar-cut $\langle t|e\rangle$ is more general on the right premiss, as e can be a left-introduction, whereas a named expression $a(H)$ is more general on its left (and unique) premiss, as H can be an elimination. Parigot's named terms can be seen as the common case of covar-cuts and the construction $a(H)$: such case reads $a(t) = \langle t|a\rangle$ ($e = a$ is not a left introduction) and $a(M) = a(\mathsf{h}(M))$ ($H = \mathsf{h}(M)$ is not an elimination). See Figure 9 for a summary of this discussion.[17]

4.3. Contexts vs co-terms

The isomorphism $\overline{\lambda}\mu\tilde{\mu} \cong \underline{\lambda}\mu\mathsf{let}$ to be proved below holds between the respective reduction relations. Since many of the reduction rules of $\underline{\lambda}\mu\mathsf{let}$ are defined with the help of contexts, we have to understand better how mappings Θ and Ψ deal with such syntactic concept.

Let *Contexts* (resp. *HoleExpressions*, *Statements*) be a shorthand for the set $\underline{\lambda}\mu\mathsf{let} - Contexts$ (resp. $\underline{\lambda}\mu\mathsf{let}$-*HoleExpressions*, $\underline{\lambda}\mu\mathsf{let}$-*Statements*). Let *CoTerms* be a shorthand for the set $\overline{\lambda}\mu\tilde{\mu}$-*CoTerms*.

There is an algebra $\mathbb{E}$ with carrier *Contexts* and the following operations: one constant $a([\_])$ for each a; one constant $\mathsf{let}\, x = [\_] \,\mathsf{in}\, S$ for each S; and, for each N, a unary operation that maps $\mathcal{E}$ to $\mathcal{E}[[\_]N]$. In fact, each $\mathcal{E} \in Contexts$ is generated by $\mathbb{E}$ in a unique way[18]. In particular, contexts can be defined inductively as follows:

$$\mathcal{E} ::= a([\_]) \mid \mathsf{let}\, x = [\_] \,\mathsf{in}\, S \mid \mathcal{E}[[\_]N] \tag{10}$$

On the other hand, each context $\mathcal{E}$ corresponds naturally with a function of type $HoleExpressions \to Statements$; it is the function that sends H to $\mathcal{E}[H]$. We call *semantic context* a function of type $HoleExpressions \to Statements$ in the range of this correspondence. In fact, the described correspondence is 1-1 and so a bijection between *Contexts* and the set of semantics contexts.

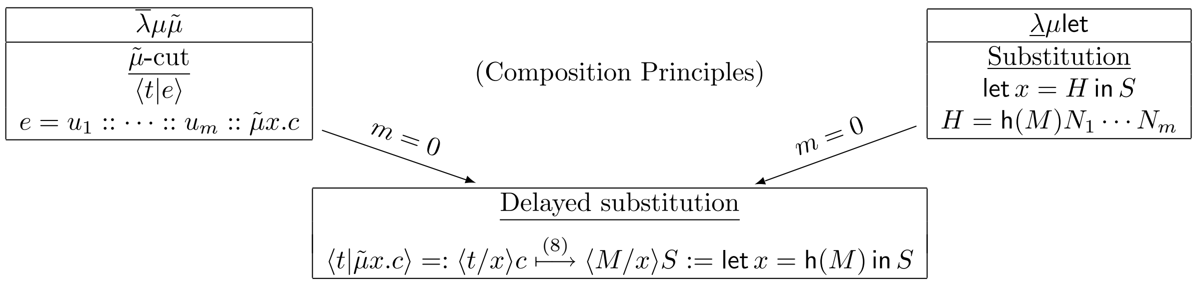
Let e be a co-term of $\overline{\lambda}\mu\tilde{\mu}$ and consider $\Theta(\_, e) : HoleExpressions \to Statements$. Is $\Theta(\_, e)$ a semantic context? Let us introduce the auxiliary mapping $\hat{\Theta}$ given in the left half of Figure 10. By a straightforward induction on e one proves that

$$(\hat{\Theta}e)[H] = \Theta(H, e) \ . \tag{11}$$

[17]In intuitionistic logic [12], cuts correspond only to substitution. The idea of delayed substitution as a common particular case of cut and substitution is also found in *op. cit.*

[18]For this reason, there is the possibility, not followed in the present paper, of identifying contexts with "contextuals" [13], that is, the closed Σ-terms (in the sense of universal algebra), where Σ is the signature of the algebra $\mathbb{E}$.

Figure 9: Composition and naming principles


$\bar{\lambda}\mu\tilde{\mu}$
$\underline{\tilde{\mu}\text{-cut}}$
$\langle t|e\rangle$
$e = u_1 :: \cdots :: u_m :: \tilde{\mu}x.c$
(Composition Principles)
$\underline{\lambda}\mu\mathsf{let}$
$\underline{\text{Substitution}}$
$\mathsf{let}\, x = H \,\mathsf{in}\, S$
$H = \mathsf{h}(M)N_1 \cdots N_m$
$m = 0$
$m = 0$
$\underline{\text{Delayed substitution}}$
$\langle t|\tilde{\mu}x.c\rangle =: \langle t/x\rangle c \xmapsto{(8)} \langle M/x\rangle S := \mathsf{let}\, x = \mathsf{h}(M) \,\mathsf{in}\, S$

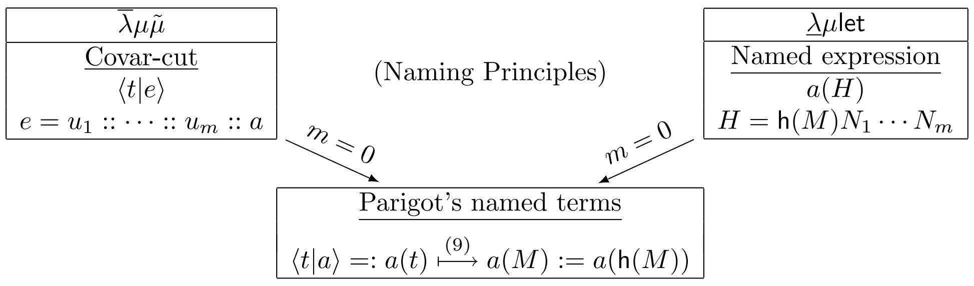

$\bar{\lambda}\mu\tilde{\mu}$
$\underline{\text{Covar-cut}}$
$\langle t|e\rangle$
$e = u_1 :: \cdots :: u_m :: a$
(Naming Principles)
$\underline{\lambda}\mu\mathsf{let}$
$\underline{\text{Named expression}}$
$a(H)$
$H = \mathsf{h}(M)N_1 \cdots N_m$
$m = 0$
$m = 0$
$\underline{\text{Parigot's named terms}}$
$\langle t|a\rangle =: a(t) \xmapsto{(9)} a(M) := a(\mathsf{h}(M))$

Figure 10: $\hat{\Theta} : CoTerms \to Contexts$ and $\hat{\Psi} : Contexts \to CoTerms$

$$\boxed{\begin{array}{rcl} \hat{\Theta}(a) & = & a([\_]) \\ \hat{\Theta}(\tilde{\mu}x.c) & = & \mathsf{let}\, x = [\_] \,\mathsf{in}\, \Theta c \\ \hat{\Theta}(u :: e) & = & (\hat{\Theta}e)[[\_]\Theta u] \end{array}} \qquad \boxed{\begin{array}{rcl} \hat{\Psi}(a([\_])) & = & a \\ \hat{\Psi}(\mathsf{let}\, x = [\_] \,\mathsf{in}\, S) & = & \tilde{\mu}x.\Psi S \\ \hat{\Psi}(\mathcal{E}[[\_]N]) & = & \Psi N :: \hat{\Psi}\mathcal{E} \end{array}}$$

Figure 11: Correspondence of reduction rules

R in $\overline{\lambda}\mu\tilde{\mu}$	R' in $\underline{\lambda}\mu\mathsf{let}$
$\beta_{\supset}$	$\beta_{\supset}$
$\sigma_{\tilde{\mu}}$	σ_{let}
σ_{μ}	σ_{μ}
$\eta_{\supset}$	$\eta_{\supset}$
$\eta_{\tilde{\mu}}$	η_{let}
η_{μ}	η_{μ}

Hence, not only $\Theta(\_, e)$ is a semantic context, it is the semantic context corresponding to the context $\hat{\Theta}e$.

The auxiliary mapping $\hat{\Theta}$ is useful in the proof of $\overline{\lambda}\mu\tilde{\mu} \cong \underline{\lambda}\mu\mathsf{let}$ because of (11), by which a statement $\Theta(H, e)$ is analyzed as H filled in the hole of context $\hat{\Theta}e$. The inverse of mapping $\hat{\Theta}$ is also useful, and given in the right half of Figure 10. Clearly:

$$\hat{\Psi}\hat{\Theta}e = e \qquad \hat{\Theta}\hat{\Psi}\mathcal{E} = \mathcal{E} \ . \tag{12}$$

4.4. Isomorphism and corollaries

Theorem 2 ($\overline{\lambda}\mu\tilde{\mu} \cong \underline{\lambda}\mu\mathsf{let}$). *The mappings Θ and Ψ are inverse isomorphisms of reduction relations at the levels of terms and commands/statements. More precisely, let R be a reduction rule of $\overline{\lambda}\mu\tilde{\mu}$ and R' be the corresponding reduction rule of $\underline{\lambda}\mu\mathsf{let}$ according to Figure 11*[19]*. Then:*

1. *For terms:*
   (a) *$t \to_R t'$ in $\overline{\lambda}\mu\tilde{\mu}$ iff $\Theta t \to_{R'} \Theta t'$ in $\underline{\lambda}\mu\mathsf{let}$.*
   (b) *$M \to_{R'} M'$ in $\underline{\lambda}\mu\mathsf{let}$ iff $\Psi M \to_R \Psi M'$ in $\overline{\lambda}\mu\tilde{\mu}$.*
2. *For co-terms/hole-expressions:*
   (a) *$e \to_R e'$ in $\overline{\lambda}\mu\tilde{\mu}$ only if, for all H, $\Theta(H, e) \to_{R'} \Theta(H, e')$ in $\underline{\lambda}\mu\mathsf{let}$;*
   (b) *$H \to_{R'} H'$ in $\underline{\lambda}\mu\mathsf{let}$ only if, for all e, $\Psi(H, e) \to_R \Psi(H', e)$ in $\overline{\lambda}\mu\tilde{\mu}$;*
3. *For commands/statements:*
   (a) *$c \to_R c'$ in $\overline{\lambda}\mu\tilde{\mu}$ iff $\Theta c \to_{R'} \Theta c'$ in $\underline{\lambda}\mu\mathsf{let}$.*
   (b) *$S \to_{R'} M'$ in $\underline{\lambda}\mu\mathsf{let}$ iff $\Psi S \to_R \Psi S'$ in $\overline{\lambda}\mu\tilde{\mu}$.*

[19]As said before, the case $\eta_{\supset}$ is postponed to Section 6.

Proof: The "if" statements in 1. follow from the "only if" statements in 1. and bijection. Similarly, the "if" statements in 3. follow from the "only if" statements in 3. and bijection. So, it suffices to prove the "only if" statements.

The "only if" statements 1(a), 2(a) and 3(a) use the following properties of Θ:

(i) $\Theta([e/a]t) = [\hat{\Theta}e/a]\Theta t$ and $\Theta([e/a]c) = [\hat{\Theta}e/a]\Theta c$;

(ii) $\Theta(\langle u/x\rangle c) = \langle \Theta u/x\rangle \Theta c$;

(iii) $\Theta([u/x]t) = [\Theta u/x]\Theta t$ and $\Theta([u/x]c) = [\Theta u/x]\Theta c$;

(iv) $H \to H' \Rightarrow \Theta(H, e) \to \Theta(H', e)$.

(i) is proved together with $\Theta([\hat{\Theta}e/a]H, [e/a]e') = [\hat{\Theta}e/a]\Theta(H, e')$, for all H. The proof is a simultaneous induction on t, c and e'. (ii) is proved by a simple calculation (this is actually (8)). (iii) is proved together with $\Theta([\Theta t/x]H, [t/x]e) = [\Theta t/x]\Theta(H, e)$, for all H. The proof is a simultaneous induction on t, c and e. (iv) is proved by induction on e.

The "only if" statements 1(a), 2(a), and 3(a) are proved by simultaneous induction on $t \to_R t'$ and $e \to_R e'$ and $c \to_R c'$. Here are the base cases:

Case $\beta_\supset$: $\langle \lambda x.t | u :: e\rangle \to \langle u/x\rangle\langle t|e\rangle$.

$$\begin{array}{lll}
 & \Theta(\langle \lambda x.t | u :: e\rangle) & \\
= & \Theta(\mathsf{h}(\lambda x.\Theta t)\Theta u, e) & \text{(by def. of } \Theta) \\
= & (\hat{\Theta}e)[\mathsf{h}(\lambda x.\Theta t)\Theta u] & \text{(by (11))} \\
\to_{\beta_\supset} & \langle \Theta u/x\rangle(\hat{\Theta}e)[\mathsf{h}(\Theta t)] & \\
= & \langle \Theta u/x\rangle\Theta(\mathsf{h}(\Theta t), e) & \text{(by (11))} \\
= & \langle \Theta u/x\rangle\Theta\langle t|e\rangle & \text{(by def. of } \Theta) \\
= & \Theta(\langle u/x\rangle\langle t|e\rangle) & \text{(by (ii))}
\end{array}$$

Case σ_μ: $\langle e/a\rangle c \to [e/a]c$.

$$\begin{array}{rcll}
\Theta\langle e/a\rangle c & = & \Theta(\mathsf{h}(\mu a.\Theta c), e) & \text{(by def. of } \Theta) \\
 & = & (\hat{\Theta}e)[\mathsf{h}(\mu a.\Theta c)] & \text{(by (11))} \\
 & \to_{\sigma_\mu} & [\hat{\Theta}e/a]\Theta c & \\
 & = & \Theta([e/a]c) & \text{(by (i))}
\end{array}$$

Case $\sigma_{\tilde{\mu}}$: $\langle t/x\rangle c \to [t/x]c$.

$$\begin{array}{rcll}
\Theta(\langle t/x\rangle c) & = & \langle \Theta t/x\rangle \Theta c & \text{(by (ii))} \\
 & \to_{\sigma_{\mathsf{let}}} & [\Theta t/x]\Theta c & \\
 & = & \Theta([t/x]c) & \text{(by (iii))}
\end{array}$$

Case η_μ: $\mu a.\langle t|a\rangle \to t$, with $a \notin t$.

$$\begin{array}{rcll}
\Theta(\mu a.\langle t|a\rangle) & = & \mu a.a(\mathsf{h}(\Theta t)) & \text{(by def. of } \Theta) \\
 & \to_{\eta_\mu} & \Theta t & (a \notin \Theta t)
\end{array}$$

Case $\eta_{\tilde{\mu}}$: $\tilde{\mu}x.\langle x|e\rangle \to e$, with $x \notin e$.

$$\begin{array}{lll} & \Theta(H, \tilde{\mu}x.\langle x|e\rangle) & \\ = & \mathsf{let}\, H = x \,\mathsf{in}\, \Theta(\mathsf{h}(x), e) & \text{(by def. of } \Theta) \\ = & \mathsf{let}\, H = x \,\mathsf{in}\, (\hat{\Theta}e)[\mathsf{h}(x)] & \text{(by (11))} \\ \to_{\eta_{\tilde{\mu}}} & (\hat{\Theta}e)[H] & (x \notin \Theta e) \\ = & \Theta(H, e) & \text{(by (11))} \end{array}$$

Most of the inductive cases follow simply by IH. There are two, however, that use (iv). We illustrate one example.

Suppose $u :: e \to u' :: e$, with $u \to u'$. Let H be arbitrary. We want $\Theta(H, u :: e) \to \Theta(H, u' :: e)$. By IH, $\Theta u \to \Theta u'$. So $H\Theta u \to H\Theta u'$. By (iv), $\Theta(H\Theta u, e) \to \Theta(H\Theta u', e)$. We are done, since $\Theta(H, u :: e) = \Theta(H\Theta u, e)$ and $\Theta(H, u' :: e) = \Theta(H\Theta u', e)$.

The "only if" statements 1(b), 2(b), and 3(b) use the following properties of Ψ:

(i) $\Psi([\mathcal{E}/a]M) = [\hat{\Psi}\mathcal{E}/a]\Psi M$ and $\Psi([\mathcal{E}/a]S) = [\hat{\Psi}\mathcal{E}/a]\Psi S$;

(ii) $\Psi(\langle N/x\rangle S) = \langle \Psi N/x\rangle \Psi S$;

(iii) $\Psi([N/x]M) = [\Psi N/x]\Psi M$ and $\Psi([N/x]S) = [\Psi N/x]\Psi S$.

(iv) $e \to e' \Rightarrow \Psi(H, e) \to \Psi(H, e')$

(i) is proved together with $\Psi([\mathcal{E}/a]H, [\hat{\Psi}\mathcal{E}/a]e) = [\hat{\Psi}\mathcal{E}/a]\Psi(H, e)$, for all e. The proof is a simultaneous induction on M, S and H. (ii) is proved by a simple calculation. (iii) is proved together with $\Psi([N/x]H, [\Psi N/x]e) = [\Psi N/x]\Psi(H, e)$, for all e. The proof is a simultaneous induction on M, S and H. (iv) is proved by induction on H.

We also need the following remark:

$$e = \hat{\Psi}\mathcal{E} \Rightarrow \Psi(\mathcal{E}[H]) = \Psi(H, e) \ . \tag{13}$$

Indeed, if $e = \hat{\Psi}\mathcal{E}$, then $\mathcal{E} = \hat{\Theta}e$. Hence, $\Psi(\mathcal{E}[H]) = \Psi((\hat{\Theta}e)[H]) \overset{(11)}{=} \Psi\Theta(H, e) = \Psi(H, e)$.

The "only if" statements 1(b), 2(b), and 3(b) are proved by simultaneous induction on $M \to_R M'$ and $H \to_R H'$ and $S \to_R S'$. Here are the base cases:

Case $\beta_\supset$: $\mathcal{E}[\mathsf{h}(\lambda x.M)N] \to \langle N/x\rangle\mathcal{E}[\mathsf{h}(M)]$). Let $e = \hat{\Psi}\mathcal{E}$.

$$\begin{array}{lll} & \Psi(\mathcal{E}[\mathsf{h}(\lambda x.M)N]) & \\ = & \Psi(\mathsf{h}(\lambda x.M)N, e) & \text{(by (13))} \\ = & \langle \lambda x.\Psi M | \Psi N :: e\rangle & \text{(by def. of } \Psi) \\ \to_{\beta_\supset} & \langle \Psi N/x\rangle \langle \Psi M|e\rangle & \\ = & \langle \Psi N/x\rangle \Psi(\mathsf{h}(M), e) & \text{(by def. of } \Psi) \\ = & \langle \Psi N/x\rangle \Psi(\mathcal{E}[\mathsf{h}(M)]) & \text{(by (13))} \\ = & \Psi(\langle N/x\rangle \mathcal{E}[\mathsf{h}(M)]) & \text{(by (ii))} \end{array}$$

Case σ_μ: $\mathcal{E}[\mathsf{h}(\mu a.S)] \to [\mathcal{E}/a]S$. Let $e = \hat{\Psi}\mathcal{E}$.

$$\begin{array}{llll} \Psi(\mathcal{E}[\mathsf{h}(\mu a.S)] & = & \Psi(\mathsf{h}(\mu a.S), e) & \text{(by (13))} \\ & = & \langle \mu a.\Psi S|e\rangle & \text{(by def. of } \Psi) \\ & \to_{\sigma_\mu} & [e/a]\Psi S & \\ & = & [\hat{\Psi}\mathcal{E}/a]\Psi S & \text{(by assumption)} \\ & = & \Psi([\mathcal{E}/a]S) & \text{(by (i))} \end{array}$$

Case σ_{let}: $\langle N/x\rangle S \to [N/x]S$.

$$\begin{array}{rcll} \Psi(\langle N/x\rangle S) & = & \langle \Psi N/x\rangle \Psi S & \text{(by (ii))} \\ & \to_{\sigma_{\tilde{\mu}}} & [\Psi N/x]\Psi S & \\ & = & \Psi([N/x]S) & \text{(by (iii))} \end{array}$$

Case η_μ: $\mu a.a(\mathsf{h}(M)) \to M$, with $a \notin M$.

$$\begin{array}{rcll} \Psi(\mu a.a(\mathsf{h}(M))) & = & \mu a.\langle \Psi M|a\rangle & \text{(by def. of } \Psi) \\ & \to_{\eta_\mu} & \Psi M & (a \notin \Psi M) \end{array}$$

Case $\eta_{\tilde{\mu}}$: $\mathsf{let}\, H = x \,\mathsf{in}\, \mathcal{E}[\mathsf{h}(x)] \to \mathcal{E}[H]$, with $x \notin \mathcal{E}$. Let $e = \hat{\Psi}\mathcal{E}$.

$$\begin{array}{rll} & \Psi(\mathsf{let}\, H = x \,\mathsf{in}\, \mathcal{E}[\mathsf{h}(x)]) & \\ = & \Psi(H, \tilde{\mu}x.\Psi(\mathcal{E}[\mathsf{h}(x)])) & \text{(by def. of } \Psi) \\ = & \Psi(H, \tilde{\mu}x.\Psi(\mathsf{h}(x), e)) & \text{(by (13))} \\ = & \Psi(H, \tilde{\mu}x.\langle x|e\rangle) & \text{(by def. of } \Psi) \\ \to_{\eta_{\tilde{\mu}}} & \Psi(H, e) & \text{(by (iv))} \\ = & \Psi(\mathcal{E}[H]) & \text{(by (13))} \end{array}$$

Most of the inductive cases follow simply by IH. There are two, however, that use (iv). We illustrate one example.

Suppose $\mathsf{let}\, x = H \,\mathsf{in}\, S \to \mathsf{let}\, x = H \,\mathsf{in}\, S'$, with $S \to S'$. We want $\Psi(\mathsf{let}\, x = H \,\mathsf{in}\, S) \to \Psi(\mathsf{let}\, x = H \,\mathsf{in}\, S')$. By IH $\Psi S \to \Psi S'$. So $\tilde{\mu}x.\Psi S \to \tilde{\mu}x.\Psi S'$. By (iv) $\Psi(H, \tilde{\mu}x.\Psi S) \to \Psi(H, \tilde{\mu}x.\Psi S')$. We are done since $\Psi(\mathsf{let}\, x = H \,\mathsf{in}\, S) = \Psi(H, \tilde{\mu}x.\Psi S)$ and $\Psi(\mathsf{let}\, x = H \,\mathsf{in}\, S') = \Psi(H, \tilde{\mu}x.\Psi S')$. $\square$

Corollary 1 (SN). *Every typable expression of $\underline{\lambda}\mu\mathsf{let}$ is $\beta_{\supset}\sigma_{\mathsf{let}}\sigma_\mu\eta_\mu\eta_{\mathsf{let}}$-SN.*

Proof: A consequence of the facts: SN holds of $\overline{\lambda}\mu\tilde{\mu}$ [27], Ψ is sound, and $\overline{\lambda}\mu\tilde{\mu} \cong \underline{\lambda}\mu\mathsf{let}$. Let us see the case of a typable hole expression H (the cases of a term or statement being similar and slightly simpler). Suppose there is an infinite reduction sequence from H. Suppose $\Gamma \rhd H : A|\Delta$. Let a be a fresh co-variable. Then $\Gamma \rhd H : A|\Delta, a : A$ (since weakening is admissible) and $\Gamma|a : A \vdash \Delta, a : A$. By part 2(b) of Theorem 2, there is an infinite reduction sequence from $\Psi(H, a)$. But $\Psi(H, a)$ is a typable command, by soundness of Ψ. This contradicts SN of $\overline{\lambda}\mu\tilde{\mu}$. $\square$

Corollary 2 (Subject reduction). *In $\underline{\lambda}\mu\mathsf{let}$, subject reduction holds of terms, hole expression, and statements.*

Proof: A consequence of the facts: subject reduction holds of $\overline{\lambda}\mu\tilde{\mu}$, Θ and Ψ are sound, and $\overline{\lambda}\mu\tilde{\mu} \cong \underline{\lambda}\mu\mathsf{let}$. For reduction on terms or statements, the proof is straightforward. Let us see the case of reduction on hole expressions.

Suppose $H \to H'$ and $\Gamma \rhd H : A|\Delta$. Let a be a fresh co-variable. Then $\Gamma \rhd H : A|\Delta, a : A$ (since weakening is admissible) and $\Gamma|a : A \vdash \Delta, a : A$. By part 2(b) of Theorem 2, $\Psi(H, a) \to \Psi(H', a)$. By soundness of Ψ, $\Psi(H, a) : (\Gamma \vdash \Delta, a : A)$ and, by subject reduction of $\overline{\lambda}\mu\tilde{\mu}$, $\Psi(H', a) : (\Gamma \vdash \Delta, a : A)$. By soundness of Θ, $\Theta\Psi(H', a) : (\Gamma \vdash \Delta, a : A)$, so Proposition 2 gives $\Theta(H', a) : (\Gamma \vdash \Delta, a : A)$, hence $a(H') : (\Gamma \vdash \Delta, a : A)$. By inversion of *Pass* we get $\Gamma \rhd H' : A|\Delta, a : A$, and admissibility of strengthening yields $\Gamma \rhd H' : A|\Delta$. $\square$

5. Call-by-name and call-by-value

In this section we analyze CBN and CBV in $\underline{\lambda}\mu\mathsf{let}$. We do "read-back" [7, 20] in a systematic fashion, reflecting into $\underline{\lambda}\mu\mathsf{let}$, through Θ, the definitions of CBN and CBV reductions and fragments of $\overline{\lambda}\mu\tilde{\mu}$. In the first subsection we define CBN and CBV reduction and make an overview of the results about fragments. The detailed treatment of fragments comes in the following two subsections. The study of CBN fragments of $\underline{\lambda}\mu\mathsf{let}$ leads us very close to $\lambda\mu$. We round off the section with a precise connection with this system.

5.1. Read-back

CBN and CBV reduction. In $\overline{\lambda}\mu\tilde{\mu}$, CBN and CBV reduction is defined [7] by giving priority to either $\sigma_{\tilde{\mu}}$ or σ_{μ}, respectively, in the CBN/CBV dilemma of $\overline{\lambda}\mu\tilde{\mu}$ - recall (1) in Section 2. This means:

- in CBV, the command $\langle t|\tilde{\mu}x.c\rangle$ is a $\sigma_{\tilde{\mu}}$-redex only if it is not a σ_{μ}-redex, that is, only when t is a value. This restriction of the reduction rule $\sigma_{\tilde{\mu}}$ is called *cbv* $\sigma_{\tilde{\mu}}$.
- in CBN, the command $\langle \mu a.c|e\rangle$ is a σ_{μ}-redex only if it is not a $\sigma_{\tilde{\mu}}$-redex, that is, only when e is a co-value. This restriction of the reduction rule σ_{μ} is called *cbn* σ_{μ}.

Recall the CBN-CBV dilemma of $\underline{\lambda}\mu\mathsf{let}$ in Fig. 6. In $\underline{\lambda}\mu\mathsf{let}$ we define:

- *CBV reduction:* The reduction rule $\sigma_{\tilde{\mu}}$ is restricted to $\mathsf{let}\, x = \mathsf{h}(V)\,\mathsf{in}\, S \to [V/x]S$. The restricted rule is called *cbv* $\sigma_{\tilde{\mu}}$, and can be rewritten as $\langle V/x\rangle S \to [V/x]S$.
- *CBN reduction:* The reduction rule σ_{μ} is restricted to $\mathcal{E}'[\mathsf{h}(\mu a.S)] \to [\mathcal{E}'/a]S$, where $\mathcal{E}'$ is a *co-value* in $\underline{\lambda}\mu\mathsf{let}$, that is, a context of the form $a([\_])$ or $\mathcal{E}[[\_]N]$. The restricted rule is called *cbn* σ_{μ}.

With the first restriction, the $\sigma_{\tilde{\mu}}$-reduction in Fig. 6 becomes forbidden; with the second, it is the σ_{μ}-reduction in Fig. 6 which becomes blocked.

CBN and CBV fragments. We now consider, not only restrictions to the reduction rules, but also restrictions to the sets of expressions.

$\overline{\lambda}\mu\tilde{\mu}$ contains a CBN fragment $\overline{\lambda}\mu\tilde{\mu}_{\mathsf{T}}$ (the T subsystem) and a CBV fragment $\overline{\lambda}\mu\tilde{\mu}_{\mathsf{Q}}$ (the Q subsystem), closed for CBN and CBV reduction, respectively. In addition, $\overline{\lambda}\mu\tilde{\mu}_{\mathsf{T}}$ contains itself a fragment $\overline{\lambda}\mu$ close to $\lambda\mu$, whereas $\overline{\lambda}\mu\tilde{\mu}_{\mathsf{Q}}$ contains itself a fragment $\overline{\lambda}\tilde{\mu}$ which can be "read-back" as a CBV λ-calculus. All this comes from [7] and is recalled in Appendix A.

In $\overline{\lambda}\mu\tilde{\mu}$ the T (resp. Q) subsystem is defined by requiring the right (resp. left) premiss of the left-introduction rule to be a co-value (resp.value). This is remarkably symmetric and elegant. In $\underline{\lambda}\mu\mathsf{let}$ we will have:

- *T subsystem:* obtained by restricting let-expressions to delayed substitutions.
- *Q subsystem:* obtained by restricting HN to HV.

These characterizations are perhaps not so elegant as those of $\overline{\lambda}\mu\tilde{\mu}$, but nevertheless they are either familiar (the restriction of arguments to values in the CBV case [26, 33]) or insightful (CBN case). Since $\lambda\mu$ is essentially contained in the T subsystem (as we will see), it shares with the latter the identifications that make it a CBN system. The identification of let-expressions with delayed substitutions is perceptible only from the point of view of the larger system $\underline{\lambda}\mu\mathsf{let}$; from the reduced perspective of the syntax of $\lambda\mu$ it is difficult to guess. In fact, $\lambda\mu$ (as originally defined by Parigot) follows the traditional approach in natural deduction [28] and goes even further than restricting let-expressions to delayed substitutions: it only allows meta-substitution.

In the remainder of this section, we will define and discuss CBV fragments $\underline{\lambda}\mu\mathsf{let}_\mathsf{Q}$, $\underline{\lambda}\mathsf{let}$, and CBN fragments $\underline{\lambda}\mu\mathsf{let}_\mathsf{T}$, $\underline{\lambda}\mu$ of $\underline{\lambda}\mu\mathsf{let}$. For now, we just state:

Theorem 3 (Read-back).

1. *For reduction: Θ and Ψ are inverse isomorphisms between cbv-$\sigma_{\tilde{\mu}}$-reduction in $\overline{\lambda}\mu\tilde{\mu}$ and cbv-$\sigma_{\tilde{\mu}}$-reduction in $\underline{\lambda}\mu\mathsf{let}$. Similarly for cbn-$\sigma_\mu$-reduction.*
2. *For fragments: the CBN fragments $\underline{\lambda}\mu\mathsf{let}_\mathsf{T}$, $\underline{\lambda}\mu$, and CBV fragments $\underline{\lambda}\mu\mathsf{let}_\mathsf{Q}$, $\underline{\lambda}\mathsf{let}$ of $\underline{\lambda}\mu\mathsf{let}$ are such that appropriate restrictions of Θ, Ψ establish the isomorphisms illustrated in Fig. 12.* [20]

We skip the proof. Both statements are proved by suitable adaptations of Theorem 2 (the second once the fragments are defined).

5.2. Call-by-value fragments

We now introduce the CBV natural deduction fragments that correspond to $\overline{\lambda}\mu\tilde{\mu}_\mathsf{Q}$ and $\overline{\lambda}\tilde{\mu}$.

The fragment $\underline{\lambda}\mu\mathsf{let}_\mathsf{Q}$ of $\underline{\lambda}\mu\mathsf{let}$ is defined in Fig. 13. $\underline{\lambda}\mu\mathsf{let}_\mathsf{Q}$ is obtained from $\underline{\lambda}\mu\mathsf{let}$ through two restrictions: (i) applications are constrained to the form HV, where V is a value; accordingly, contexts $\mathcal{E}$ have the restricted form of *CBV contexts*:

$$\mathcal{E}_v ::= a([\_]) \mid \mathsf{let}\, x = [\_] \,\mathsf{in}\, S \mid \mathcal{E}_v[[\_]V] \ .$$

(ii) $\sigma_{\tilde{\mu}}$ is constrained to cbv $\sigma_{\tilde{\mu}}$, which from now on we denote $\sigma_{\tilde{\mu}v}$.

The fragment $\underline{\lambda}\mathsf{let}$ of $\underline{\lambda}\mu\mathsf{let}$, smaller than $\underline{\lambda}\mu\mathsf{let}_\mathsf{Q}$, is defined in Fig. 14. $\underline{\lambda}\mathsf{let}$ is obtained from $\underline{\lambda}\mu\mathsf{let}_\mathsf{Q}$ after two steps, very much like $\overline{\lambda}\tilde{\mu}$ is obtained from $\overline{\lambda}\mu\tilde{\mu}_\mathsf{Q}$. First, μ-abstraction is removed from the syntax. This is achieved by restricting heads $\mathsf{h}(M)$ to the form $\mathsf{h}(V)$ and hiding μ-abstraction occurring under a λ through a "double abstraction" $\lambda(x,a).S$. After the first step, the class of terms can be dispensed with, and values are generated by the grammar $V ::= x \mid \lambda(x,a).S \mid \lambda x.V$. In the second step values of the form $\lambda x.V$ are dropped, since they can be recovered as $\lambda(x,a).a(\mathsf{h}(V))$, with a fresh. The double abstraction makes the $\beta_\supset$ rule generate two substitutions, which somehow compensates the absence of σ_μ. Since μ-abstraction was removed from the syntax, rule η_μ is dropped.

[20]The naming of systems is as follows. Symbols μ, T, and Q are invariant, when moving between sequent calculus and natural deduction. The remaining symbols obey the correspondence $\overline{\lambda}/\underline{\lambda}$ and $\tilde{\mu}/\mathsf{let}$.

Figure 12: CBN and CBV fragments

$$\begin{tikzcd}
& \bar{\lambda}\mu\tilde{\mu} \arrow[rr, leftrightarrow, "\Theta, \Psi"] \arrow[dl, dash, dashed] \arrow[ddd, dash, dashed] & & \underline{\lambda}\mu\mathsf{let} \arrow[dr, dash, dashed] \arrow[ddd, dash, dashed] & \\
\bar{\lambda}\mu\tilde{\mu}_{\mathsf{Q}} \arrow[rrrr, leftrightarrow, "\Theta, \Psi"] \arrow[d, dash, dashed] & & & & \underline{\lambda}\mu\mathsf{let}_{\mathsf{Q}} \arrow[d, dash, dashed] \\
\bar{\lambda}\tilde{\mu} \arrow[rrrr, leftrightarrow, "\Theta, \Psi"] & & & & \underline{\lambda}\mathsf{let} \\
& \bar{\lambda}\mu\tilde{\mu}_{\mathsf{T}} \arrow[rr, leftrightarrow, "\Theta, \Psi"] \arrow[d, dash, dashed] & & \underline{\lambda}\mu\mathsf{let}_{\mathsf{T}} \arrow[d, dash, dashed] & \\
& \bar{\lambda}\mu \arrow[rr, leftrightarrow, "\Theta, \Psi"] \arrow[drr, dash, dashed] & & \underline{\lambda}\mu \arrow[d, dash, dashed] & \\
& & & \lambda\mu &
\end{tikzcd}$$

Figure 13: $\underline{\lambda}\mu\mathsf{let}_{\mathsf{Q}}$: a CBV fragment of $\underline{\lambda}\mu\mathsf{let}$

	V	$::=$	$x \mid \lambda x.M$
	M, N	$::=$	$V \mid \mu a.S$
	H	$::=$	$\mathsf{h}(M) \mid HV$
	S	$::=$	$a(H) \mid \mathsf{let}\, x = H \,\mathsf{in}\, S$
$(\beta_{\supset})$	$\mathcal{E}_v[\mathsf{h}(\lambda x.M)V]$	$\to$	$\langle V/x\rangle \mathcal{E}_v[\mathsf{h}(M)]$
$(\sigma_{\tilde{\mu}v})$	$\langle V/x\rangle S$	$\to$	$[V/x]S$
(σ_{μ})	$\mathcal{E}_v[\mathsf{h}(\mu a.S)]$	$\to$	$[\mathcal{E}_v/a]S$
(η_{μ})	$\mu a.a(\mathsf{h}(M))$	$\to$	M, $a \notin M$
(η_{let})	$\mathsf{let}\, x = H \,\mathsf{in}\, \mathcal{E}_v[\mathsf{h}(x)]$	$\to$	$\mathcal{E}_v[H]$, $x \notin \mathcal{E}_v$

Figure 14: $\underline{\lambda}\mathsf{let}$: a CBV fragment of $\underline{\lambda}\mu\mathsf{let}$

	V	$::=$	$x \mid \lambda(x, a).S$
	H	$::=$	$\mathsf{h}(V) \mid HV$
	S	$::=$	$a(H) \mid \mathsf{let}\, x = H \,\mathsf{in}\, S$
$(\beta_{\supset})$	$\mathcal{E}_v[\mathsf{h}(\lambda(x, a).S)V]$	$\to$	$[\mathcal{E}_v/a][V/x]S$
$(\sigma_{\tilde{\mu}v})$	$\langle V/x\rangle S$	$\to$	$[V/x]S$
(η_{let})	$\mathsf{let}\, x = H \,\mathsf{in}\, \mathcal{E}_v[\mathsf{h}(x)]$	$\to$	$\mathcal{E}_v[H]$, $x \notin \mathcal{E}_v$

Discussion. There were a number of recent attempts to obtain a CBV λ-calculus (in natural deduction syntax) with a formulation validated by a good correspondence with $\overline{\lambda}\mu\tilde{\mu}$ [7, 30, 19, 20]. [21] And yet, nothing like $\underline{\lambda}\mu\mathsf{let}_\mathsf{Q}$ (the Q fragment of natural deduction) was obtained.

Curien and Herbelin give in [7] another system, named $\lambda\tilde{\mu}$, as the result of "reading $\overline{\lambda}\tilde{\mu}$ back in natural deduction style". Its syntax is defined as follows:

$$\begin{array}{rcl} V & ::= & x \mid \lambda(x,a).S \\ S & ::= & a(V\vec{V}) \mid \mathsf{let}\, x = V\vec{V}\,\mathsf{in}\, S \end{array}$$

Based on this syntax, reduction rules are given in [7] rephrasing the rules of $\overline{\lambda}\tilde{\mu}$. In addition, a mapping $\mathcal{V} : \lambda\tilde{\mu} \to \overline{\lambda}\tilde{\mu}$ is defined and claimed an isomorphism. In fact, $\underline{\lambda}\mathsf{let}$ is a formalization of $\lambda\tilde{\mu}$, and $\Psi : \underline{\lambda}\mathsf{let} \to \overline{\lambda}\tilde{\mu}$ is the formal counterpart to $\mathcal{V}$.

More generally, $\Psi : \underline{\lambda}\mu\mathsf{let}_\mathsf{Q} \to \overline{\lambda}\mu\tilde{\mu}_\mathsf{Q}$ is the isomorphism whose domain is the formal "read-back" of $\overline{\lambda}\mu\tilde{\mu}_\mathsf{Q}$. Such "read-back" is not attempted in [7] and has an admittedly unsuccessful try in [19], as follows:[22]

$$\begin{array}{rcl} M, N & ::= & x \mid \lambda x.M \mid \mu a.S \\ A & ::= & \mathsf{h}(M) \mid AN \mid \mathsf{let}\, x = A\,\mathsf{in}\, A \\ S & ::= & a(A) \end{array}$$

Relatively to $\underline{\lambda}\mu\mathsf{let}$, this syntax has a remarkable difference: the placement of let-expressions. We believe this causes even bijection to $\overline{\lambda}\mu\tilde{\mu}_\mathsf{Q}$ to fail, as reported in [19].

Finally, the system in [20] places applications and let-expressions in the class of terms. As a result, there is in this system a set of "structural" reduction rules and some "observational rules" devoted to eliminate expressions which in $\underline{\lambda}\mu\mathsf{let}$ (and therefore in $\overline{\lambda}\mu\tilde{\mu}$) are already ruled out by the organization of the syntactic classes. For instance, one such rule is the "associativity" of let-expressions, that reduces a let whose actual parameter is another let. Such expression does not exist in $\underline{\lambda}\mu\mathsf{let}$.

5.3. Call-by-name fragments

We now introduce the CBN natural deduction fragments that correspond to $\overline{\lambda}\mu\tilde{\mu}_\mathsf{T}$ and $\overline{\lambda}\mu$.

The fragment $\underline{\lambda}\mu\mathsf{let}_\mathsf{T}$ of $\underline{\lambda}\mu\mathsf{let}$ is defined in Fig. 15. $\underline{\lambda}\mu\mathsf{let}_\mathsf{T}$ is obtained from $\underline{\lambda}\mu\mathsf{let}$ through two restrictions: (i) let-expressions are restricted to the form $\mathsf{let}\, x = \mathsf{h}(N)\,\mathsf{in}\, S$, which corresponds to a delayed substitution $\langle N/x \rangle S$; (ii) the rule σ_μ is restricted to the case

$$\mathcal{E}_n[\mathsf{h}(\mu a.S)] \to [\mathcal{E}_n/a]S \ ,$$

where *CBN contexts* $\mathcal{E}_n$ are given by

$$\mathcal{E}_n ::= a([\_]) \mid \mathcal{E}_n[[\_]N] \ .$$

[21]Rocheteau's definition of CBV [30] does not agree with Ong-Stewart's [24]. Let $M = (\mu b.S')(\mu a.S)$, let $P = \mu b.[b([\_](\mu a.S))/b]S'$ and $Q = \mu a.[a((\mu b.S')[\_])/a]S$. According to Rocheteau, there is a CBN/CBV dilemma in M, and P (resp. Q) is the CBN (resp. CBV) reduct of M. According to Ong-Stewart, there is no dilemma in M and P is the only reduct of M.

[22]It is the calculus called $\lambda^{\eta}_{\mu_{let}} - \eta_{\mu}^{let-app}$ in [19].

Figure 15: $\underline{\lambda}\mu\mathsf{let}_\mathsf{T}$: a CBN fragment of $\underline{\lambda}\mu\mathsf{let}$

$$\begin{array}{|rrcl|}
\hline
 & M, N & ::= & x \mid \lambda x.M \mid \mu a.S \\
 & H & ::= & \mathsf{h}(M) \mid HN \\
 & S & ::= & a(H) \mid \mathsf{let}\, x = \mathsf{h}(N)\,\mathsf{in}\, S \\
\hline
(\beta_\supset) & \mathcal{E}_n[\mathsf{h}(\lambda x.M)N] & \to & \langle N/x\rangle \mathcal{E}_n[\mathsf{h}(M)] \\
(\sigma_{\tilde{\mu}}) & \langle N/x\rangle S & \to & [N/x]S \\
(\sigma_{\mu n}) & \mathcal{E}_n[\mathsf{h}(\mu a.S)] & \to & [\mathcal{E}_n/a]S \\
(\eta_\mu) & \mu a.a(\mathsf{h}(M)) & \to & M,\, a \notin M \\
\hline
\end{array}$$

Figure 16: $\underline{\lambda}\mu$: a CBN fragment of $\underline{\lambda}\mu\mathsf{let}$

$$\begin{array}{|rrcl|}
\hline
 & M, N & ::= & x \mid \lambda x.M \mid \mu a.S \\
 & H & ::= & \mathsf{h}(M) \mid HN \\
 & S & ::= & a(H) \\
\hline
(\beta_\supset) & \mathsf{h}(\lambda x.M)N & \to & \mathsf{h}([N/x]M) \\
(\sigma_{\mu n}) & \mathcal{E}_n[\mathsf{h}(\mu a.S)] & \to & [\mathcal{E}_n/a]S \\
(\eta_\mu) & \mu a.a(\mathsf{h}(M)) & \to & M,\, a \notin M \\
\hline
\end{array}$$

This particular case of σ_μ is more restrictive than cbn σ_μ, and denoted $\sigma_{\mu n}$.

In $\underline{\lambda}\mu\mathsf{let}_\mathsf{T}$ rule $\eta_{\tilde{\mu}}$ is a particular case of σ and thus is omitted.

The CBN fragment $\underline{\lambda}\mu$ of $\underline{\lambda}\mu\mathsf{let}$, smaller than $\underline{\lambda}\mu\mathsf{let}_\mathsf{T}$, is defined in Fig. 16. $\underline{\lambda}\mu$ is obtained from $\underline{\lambda}\mu\mathsf{let}_\mathsf{T}$ by removing let-expressions from the syntax. Hence rule $\sigma_{\tilde{\mu}}$ is dropped, and rule $\beta_\supset$ has to compensate the absence of $\sigma_{\tilde{\mu}}$ by executing immediately the substitution it generates. Since this substitution does not have to be pulled out of the context $\mathcal{E}_n$, rule $\beta_\supset$ can be defined in a "local" style, as a relation on hole-expressions.

Discussion. Contrary to the CBV case, Curien and Herbelin do not give in [7] the system that would be the result of "reading $\overline{\lambda}\mu$ back in natural deduction style". Informally, its syntax would be defined as follows:

$$\begin{array}{rcl}
M & ::= & x \mid \lambda x.M \mid \mu a.S \\
S & ::= & a(M\vec{N})
\end{array}$$

Based on this, we could rephrase the reduction rules of $\overline{\lambda}\mu$. In fact, $\underline{\lambda}\mu$ is a formalization of the resulting system, and $\Theta : \overline{\lambda}\mu \to \underline{\lambda}\mu$ is the isomorphism between $\overline{\lambda}\mu$ and its read-back. The more general $\Theta : \overline{\lambda}\mu\tilde{\mu}_\mathsf{T} \to \underline{\lambda}\mu\mathsf{let}_\mathsf{T}$ gives the read-back of the T subsystem, which is not attempted in [7, 19].

A possible point of view is this: there is no need to pursue CBN read-back because there exists already the natural deduction system $\lambda\mu$. Maybe the absence of CBN read-back in [7] stems from this opinion. We think otherwise: as illustrated in Figure 12, $\lambda\mu$ is just one among several CBN natural deduction systems.

Figure 17: Connection with $\lambda\mu$

$$\begin{tikzcd}
\overline{\lambda}\mu \arrow[r, shift left, "\Theta"] & \underline{\lambda}\mu \arrow[l, shift left, "\Psi"] \arrow[d, shift left, "|\_|"] \\
& \lambda\mu \arrow[ul, "\mathcal{G}"] \arrow[u, shift left, "\mathsf{g}(\_)"]
\end{tikzcd}
\qquad
\begin{tikzcd}
\overline{\lambda}\mu \arrow[r, shift left, "\Theta"] & \underline{\lambda}\mu \arrow[l, shift left, "\Psi"] \arrow[d, shift left, "|\_|"] \\
& \lambda\mu \arrow[ul, "\mathcal{P}"] \arrow[u, shift left, "\mathsf{p}(\_)"]
\end{tikzcd}$$

Figure 18: Mapping $\mathcal{G} : \lambda\mu \to \overline{\lambda}\mu$

$$\boxed{\begin{array}{rcl}
\mathcal{G}(x) & = & x \\
\mathcal{G}(\lambda x.M) & = & \lambda x.\mathcal{G}M \\
\mathcal{G}(\mu a.S) & = & \mu a.\mathcal{G}(S) \\
\mathcal{G}(MN) & = & \mu a.\langle \mathcal{G}M | \mathcal{G}N :: a\rangle \ (a \text{ fresh}) \\
\mathcal{G}(a(M)) & = & \langle \mathcal{G}M | a\rangle
\end{array}}$$

5.4. Connection with $\lambda\mu$

We complete the analysis of CBN fragments by making precise the connection with $\lambda\mu$. We are going to detail the lower triangle of Figure 12, showing that $\overline{\lambda}\mu$ and $\underline{\lambda}\mu$ are variants of $\lambda\mu$. Since $\underline{\lambda}\mu$ is a fragment of $\underline{\lambda}\mu\mathsf{let}$, this completes a formal comparison of $\lambda\mu$ with the latter.

The lower triangle of Figure 12 appears in Figure 17 in two versions. One version mentions mapping $\mathcal{G} : \lambda\mu \to \overline{\lambda}\mu$, the other mapping $\mathcal{P} : \lambda\mu \to \overline{\lambda}\mu$, both due to Curien and Herbelin and defined in Figures 18 and 19[23].

Besides Θ and Ψ, the other mapping occurring in both diagrams of Figure 17 is a forgetful mapping $|\_| : \underline{\lambda}\mu \to \lambda\mu$, given in Fig. 20, that forgets the distinction between term and hole expression, and erases the marks $\mathsf{h}(\_)$. This is done with an auxiliary mapping, defined simultaneously, that maps a hole expression H to a term $|H|$.

The two remaining mappings of Figure 17 are $\mathsf{g}(\_)$ and $\mathsf{p}(\_)$, of type $\lambda\mu \to \underline{\lambda}\mu$. Like in $\underline{\lambda}\mu\mathsf{let}$ (and $\overline{\lambda}\mu\tilde{\mu}$), in $\underline{\lambda}\mu$ the ordinary application of a term to another term is not primitive. So, in a mapping of type $\lambda\mu \to \underline{\lambda}\mu$, some encoding of MN is needed. The obvious choice is $\mu a.a(\mathsf{h}(M)N)$, with a fresh. The mapping $\mathsf{g}(\_) : \lambda\mu \to \underline{\lambda}\mu$ given in Fig. 21 is based on this idea. However, in the case of multiple application, for instance MN_1N_2, this definition can be optimized as follows:

$$\mu a_2.a_2(\mathsf{h}(\mu a_1.a_1(\mathsf{h}(M)N_1))N_2) \quad = \quad \mu a_2.\mathcal{E}_n[\mathsf{h}(\mu a_1.a_1(\mathsf{h}(M)N_1))] \quad \text{with } \mathcal{E}_n = a_2([\_]N_2)$$

[23]The mapping we call $\mathcal{G}$ is named $(\_)^n$ in [7] and $(\_)^>$ in [19]. The mapping we call $\mathcal{P}$ is named $(\_)^{\mathcal{N}}$ in [7] and $(\_)^{\mathsf{N}}$ in [19]. $\mathcal{G}$ *"amounts to translate natural deduction into sequent calculus"* [7], according to an idea that goes back to Gentzen [15]. As recognized in [19], $\mathcal{P}$ adapts another idea of translation of natural deduction into sequent calculus, due to Prawitz [28]. The intuitionistic version of $\mathcal{P}$ was studied and proved to be an isomorphism in [9, 10].

Figure 19: Mapping $\mathcal{P} : \lambda\mu \to \overline{\lambda}\mu$

$$
\begin{array}{rcl}
\mathcal{P}(x) & = & x \\
\mathcal{P}(\lambda x.M) & = & \lambda x.\mathcal{P}M \\
\mathcal{P}(\mu a.S) & = & \mu a.\mathcal{P}(S) \\
\mathcal{P}(MN) & = & \mu a.\mathcal{P}(M, \mathcal{P}(N) :: a) \text{ (a fresh)}
\end{array}
$$

$$
\begin{array}{rcl|rcl}
\mathcal{P}(a(M)) & = & \mathcal{P}(M, a) & \mathcal{P}(MN, E) & = & \mathcal{P}(M, \mathcal{P}(N) :: E) \\
 & & & \mathcal{P}(M, E) & = & \langle \mathcal{P}(M) | E \rangle \text{ (M not application)}
\end{array}
$$

Figure 20: Mapping $|\_| : \underline{\lambda}\mu \to \lambda\mu$

$$
\begin{array}{rcl|rcl}
|x| & = & x & |a(H)| & = & a(|H|) \\
|\lambda x.M| & = & \lambda x.|N| & |HN| & = & |H||N| \\
|\mu a.S| & = & \mu a.|S| & |\mathsf{h}(M)| & = & |M|
\end{array}
$$

$$
\begin{array}{cl}
\to_{\sigma_{\mu n}} & \mu a_2.[\mathcal{E}_n/a_1](a_1(\mathsf{h}(M)N_1)) \\
= & \mu a_2.a_2(\mathsf{h}(M)N_1N_2)
\end{array}
$$

This suggests to translate the inner application MN_1, not as a term, but instead as the hole expression $\mathsf{h}(M)N_1$ - thereby sparing some η_μ-expansions. The mapping $\mathsf{p}(\_) : \lambda\mu \to \underline{\lambda}\mu$ given in Fig. 22 is based on this idea. Three mappings are defined by simultaneous recursion: one sending a term M to a term $\mathsf{p}(M)$; another sending a statement S to a statement $\mathsf{p}(S)$; yet another sending a term M to a hole expression $\mathsf{p}(M)'$.

The alternative between $\mathsf{g}(\_)$ and $\mathsf{p}(\_)$ is related to the alternative between $\mathcal{G}$ and $\mathcal{P}$. As the next result shows, $\mathsf{g}(M)$ and $\mathcal{G}(M)$ are related by Θ/Ψ, and similarly for $\mathsf{p}(M)$ and $\mathcal{P}(M)$.

Theorem 4 (Connection with $\lambda\mu$). *Let $|\underline{\lambda}\mu| \subset \lambda\mu$ be the range of $|\_|$ and consider the following form of η_μ-expansion in $\lambda\mu$:*

Figure 21: Mapping $\mathsf{g}(\_) : \lambda\mu \to \underline{\lambda}\mu$

$$
\begin{array}{rcl}
\mathsf{g}(x) & = & x \\
\mathsf{g}(\lambda x.M) & = & \lambda x.\mathsf{g}(M) \\
\mathsf{g}(\mu a.S) & = & \mu a.\mathsf{g}(S) \\
\mathsf{g}(MN) & = & \mu a.a(\mathsf{h}(\mathsf{g}(M))\mathsf{g}(N)) \text{ (a fresh)} \\
\mathsf{g}(a(M)) & = & a(\mathsf{h}(\mathsf{g}(M)))
\end{array}
$$

Figure 22: Mapping $\mathsf{p}(\_) : \lambda\mu \to \underline{\lambda\mu}$

$\mathsf{p}(x) = x$	$\mathsf{p}(a(M)) = a(\mathsf{p}(M)')$
$\mathsf{p}(\lambda x.M) = \lambda x.\mathsf{p}(M)$	
$\mathsf{p}(\mu a.S) = \mu a.\mathsf{p}(S)$	
$\mathsf{p}(MN) = \mu a.a(\mathsf{p}(M)'\mathsf{p}(N))$ (a fresh)	$\mathsf{p}(MN)' = \mathsf{p}(M)'\ \mathsf{p}(N)$
	$\mathsf{p}(M)' = \mathsf{h}(\mathsf{p}(M))$ (M not application)

$$MN \to \mu a.a(MN)\ \textit{(a fresh)}\ . \tag{14}$$

Then:

1. $\Psi \circ \mathsf{g}(\_) = \mathcal{G}$ *and* $\Theta \circ \mathcal{G} = \mathsf{g}(\_)$ *and* $\Psi \circ \mathsf{p}(\_) = \mathcal{P}$ *and* $\Theta \circ \mathcal{P} = \mathsf{p}(\_)$.
2. *For all* $M \in \lambda\mu$, *both* $|\mathsf{g}(M)|$ *and* $|\mathsf{p}(M)|$ *result from* M *by* η_μ*-expansions (14) .*
3. *For all* $M \in \underline{\lambda\mu}$, $\mathsf{p}(|M|) = M$.
4. *For all* $t \in \overline{\lambda\mu}$, $\mathcal{P}|\Theta t| = t$.
5. *The sets of terms* $\overline{\lambda\mu}$, $\underline{\lambda\mu}$, *and* $|\underline{\lambda\mu}|$ *are in bijective correspondence.*
6. *The restriction of* $\mathcal{P}$ *to* $|\underline{\lambda\mu}|$ *is a bijection.*

Proof: 1. One proves: (i) $\mathcal{G}(M) = \Psi(\mathsf{g}(M))$ and $\mathcal{P}(M) = \Psi(\mathsf{p}(M))$ and $\mathcal{P}(M, E) = \Psi(\mathsf{p}(M)', E)$, (all E); (ii) $\mathcal{G}(S) = \Psi(\mathsf{g}(S))$ and $\mathcal{P}(S) = \Psi(\mathsf{p}(S))$. The proof is a simultaneous induction on $M \in \lambda\mu$ and $S \in \lambda\mu$. Hence, if $M \in \lambda\mu$, then $\Theta\mathcal{G}(M) = \Theta\Psi(\mathsf{g}(M)) = \mathsf{g}(M)$. The latter equality is by $\overline{\lambda\mu} \cong \underline{\lambda\mu}$ (Theorem 3). Similarly for $\mathcal{P}$ and $\mathsf{p}(\_)$ and for $S \in \lambda\mu$.

2. One proves: (i) $|\mathsf{g}(M)| \to^*_{\eta_\mu} M$ and $|\mathsf{p}(M)| \to^*_{\eta_\mu} M$ and $|\mathsf{p}(M)'| \to^*_{\eta_\mu} M$; (ii) $|\mathsf{g}(S)| \to^*_{\eta_\mu} S$ and $|\mathsf{g}(S)| \to^*_{\eta_\mu} S$. The proof is a simultaneous induction on $M \in \lambda\mu$ and $S \in \lambda\mu$.

3. One proves: (i) $\mathsf{p}(|M|) = M$ and $\mathsf{p}(|M|)' = \mathsf{h}(M)$; (ii) $\mathsf{p}(|S|) = S$; (iii) $\mathsf{p}(|H|)' = H$. The proof is a simultaneous induction on $M \in \underline{\lambda\mu}$, $S \in \underline{\lambda\mu}$, and $H \in \underline{\lambda\mu}$.

4. $\mathcal{P}|\Theta t| \overset{1.}{=} \Psi\mathsf{p}(|\Theta t|) \overset{3.}{=} \Psi\Theta t = t$. The latter equality is by $\overline{\lambda\mu} \cong \underline{\lambda\mu}$.

5. On the one hand, $\overline{\lambda\mu} \cong \underline{\lambda\mu}$. On the other hand, statement 3. says that mapping $|\_|$ is an injection, hence a bijection between the sets $\underline{\lambda\mu}$ and $|\underline{\lambda\mu}|$.

6. Statement 3. implies that $|\underline{\lambda\mu}|$ is the range of the endo-function $F : \lambda\mu \to \lambda\mu$ defined by $F = |\_| \circ \mathsf{p}(\_)$. Statement 4. says that $|\_| \circ \Theta$ is an injection as well, hence a bijection between $\overline{\lambda\mu}$ and another subset of $\lambda\mu$, namely the range of the endo-function $|\_| \circ \Theta \circ \mathcal{P}$ of type $\lambda\mu \to \lambda\mu$. But this endo-function is the same as the previous F, since $|\Theta\mathcal{P}M| = |\mathsf{p}(M)|$, by statement 1. So $|\_| \circ \Theta$ is a bijection between $\overline{\lambda\mu}$ and $|\underline{\lambda\mu}|$ whose inverse, by statement 4., is the restriction of $\mathcal{P}$ to $|\underline{\lambda\mu}|$. $\square$

Discussion. It is $\mathcal{P}$ - not $\mathcal{G}$ - that yields bijection between $|\underline{\lambda\mu}| \subset \lambda\mu$ and $\overline{\lambda\mu}$. Curien and Herbelin argue in [7] that this bijection can be turned into an isomorphism. Of course, the domain of such bijection is not described in [7] as $|\underline{\lambda\mu}|$, it is described rather as a subset of $\lambda\mu$ whose elements are in some η_μ-extended form[24]. Yet it is unpleasant to base a calculus on such subset: is it closed for β or μ reduction? The answer is not immediate. Certainly the subset is not closed for η_μ-reduction, as we do not want to reverse the expansions (14). In addition, for isomorphism to hold, one has to split the $\sigma_{\mu n}$ rule of $\overline{\lambda}\mu$ into two new rules that mimic the stepwise way of proceeding of the μ rule of $\lambda\mu$ [7, 19].

In fact, the best description of $|\underline{\lambda}\mu|$ is $\underline{\lambda}\mu$ (the description one has before the forgetful mapping erases useful distinctions); and if isomorphism with $\overline{\lambda}\mu$ is what we seek, then $\underline{\lambda}\mu$, as a calculus, is indeed isomorphic to $\overline{\lambda}\mu$ proper (no need to split rule $\sigma_{\mu n}$).

6. Semantics of $\overline{\lambda}\mu\tilde{\mu}$ and the isomorphic point of view

In the previous section, Θ was explored as a *read-back* mapping into natural deduction. In this section, Θ is analyzed as a *semantics* of $\overline{\lambda}\mu\tilde{\mu}$.

We propose a formalization of the informal semantics of $\overline{\lambda}\mu\tilde{\mu}$ expressions, based on informal concepts of "context" and "hole filling", in terms of the precise concepts of context $\mathcal{E}$ and hole filling $\mathcal{E}[H]$ provided by $\underline{\lambda}\mu\mathsf{let}$. The formalized semantics, given in Subsection 6.2, is proved sound and establishes isomorphism of reduction relations, even at the co-term/context level - something that was not achieved so far by Θ, recall Theorem 2. However, this requires some further elaboration on contexts, in order to equip them with typing and reduction rules - essentially those that contexts inherit from $\underline{\lambda}\mu\mathsf{let}$. This is preliminary work, done in Subsection 6.1.

By the end of Subsection 6.2, it becomes evident that the description of the relationship between $\overline{\lambda}\mu\tilde{\mu}$ and $\underline{\lambda}\mu\mathsf{let}$ has improved, but is incomplete - because is biased towards one of two isomorphic points of view. This is explained in Subsection 6.3. The missing element is a new syntactic concept pertaining to $\overline{\lambda}\mu\tilde{\mu}$, that we call *co-context* and introduce in Subsection 6.4.

First, the treatment of co-contexts is similar to the treatment of contexts in $\underline{\lambda}\mu\mathsf{let}$: just as a meta-linguistic device. We illustrate its use in Subsection 6.5 in the definition of $\eta_\supset$-reduction in $\overline{\lambda}\mu\tilde{\mu}$ (mirroring the use of contexts in the description of reduction rules in $\underline{\lambda}\mu\mathsf{let}$). Later, in Subsection 6.6, we equip co-contexts with the typing and reduction rules they inherit from $\overline{\lambda}\mu\tilde{\mu}$. This allows, in the final Subsection 6.7, to complete the semantics with a missing component, that connects isomorphically co-contexts and hole-expressions.

6.1. Contexts

There are three approaches to contexts of $\underline{\lambda}\mu\mathsf{let}$, in increasing order or "citizenship": the approach of $\underline{\lambda}\mu\mathsf{let}$; the intermediate approach; and the unified approach.

In $\underline{\lambda}\mu\mathsf{let}$, contexts, in spite of being formal objects derived from the syntax of $\underline{\lambda}\mu\mathsf{let}$, are not first class objects: they are not parts of other expressions, they do not contribute reduction or typing rules, or cases in the compatible closure. They are just used in the meta-language, as a device for representing statements S in the form $\mathcal{E}[H]$, a form which is particularly useful when $H = \mathsf{h}(M)$, because the hidden head is brought to the surface.

[24]A precise, but not so easy, characterization of this subset is given in [19], where its elements are called η_μ^{app}-expanded.

This is why contexts are used in the meta-linguistic description of the reduction rules of $\underline{\lambda}\mu\mathsf{let}$.

In the other extreme, there is the unified approach, according to which contexts form a primitive syntactic class of expressions, defined by simultaneous induction together with the other syntactic classes. Context constructors generate new cases for the compatible closure of reduction rules. Reduction rules (if any) and typing rules on contexts are also given simultaneously with the other reduction and typing rules of the calculus. The full integration of contexts in the primitive syntax is achieved by accepting $\mathcal{E}[H]$ as primitive syntax and the sole form of statements, realizing that the original two forms of statement in $\underline{\lambda}\mu\mathsf{let}$ are recovered as particular cases of $\mathcal{E}[H]$. Such approach was developed in full detail for the intuitionistic case in [12]. The result is a system that *unifies* sequent calculus and natural deduction (hence the name of the approach). This is far more than what we need here.

In the intermediate approach, that we are going to develop in this section, contexts remain derived objects, outside $\underline{\lambda}\mu\mathsf{let}$, but are equipped with typing and reduction rules. However, this structure is mostly inherited from $\underline{\lambda}\mu\mathsf{let}$. For instance, if $N \to N'$ in $\underline{\lambda}\mu\mathsf{let}$, then $\mathcal{E}[[\_]N] \to \mathcal{E}[[\_]N']$. The result is denoted

$$[\underline{\lambda}\mu\mathsf{let} + \mathit{Contexts}] \ .$$

This notation emphasizes that the "calculus of contexts" is disjoint from $\underline{\lambda}\mu\mathsf{let}$, in the sense that $\underline{\lambda}\mu\mathsf{let}$ is completed first, and only then contexts inherit structure from it.

Typing for contexts. We define a new typing system for contexts of $\underline{\lambda}\mu\mathsf{let}$. This system derives sequents

$$\Gamma | \mathcal{E} : A \vdash \Delta \tag{15}$$

following Curien and Herbelin's view - already recalled in Section 2.3 - that contexts are typed "on the left", with the left type being the type of the hole [7]. The novelty here is that the left type is also the type of the *hole expression* H to be filled in the hole - see Corollary 3 below.

The meaning of sequents (15) entails that the system has three typing rules:

$$\begin{gathered}
\frac{}{\Gamma | a([\_]) : A \vdash a : A, \Delta}\ (i) \qquad \frac{S : (\Gamma, x : A \vdash_{\underline{\lambda}\mu\mathsf{let}} \Delta)}{\Gamma | \mathsf{let}\, x = [\_]\, \mathsf{in}\, S : A \vdash \Delta}\ (ii) \\
\frac{\Gamma \vdash_{\underline{\lambda}\mu\mathsf{let}} N : A | \Delta \quad \Gamma | \mathcal{E} : B \vdash \Delta}{\Gamma | \mathcal{E}[[\_]N] : A \supset B \vdash \Delta}\ (iii)
\end{gathered} \tag{16}$$

The sequents with tag $\underline{\lambda}\mu\mathsf{let}$ are not premisses, but side conditions demanding the derivability of the tagged sequent. Therefore, these typing rules follow the inductive definition (10) of contexts: there are two axioms and one one-premiss inductive rule.[25]

Lemma 1. *The following rules are admissible:*

[25]The idea for typing "contexts" of the form $\mathcal{C}[[\_]N]$ is suggested in the first section of [7]. The precise formulation of these rules (i), (ii), and (iii) presupposes the precise formulation of $\underline{\lambda}\mu\mathsf{let}$.

1.

$$\frac{\Gamma|e : A \vdash \Delta}{\Gamma|\hat{\Theta}e : A \vdash \Delta}\ (a) \qquad \frac{\Gamma|\mathcal{E} : A \vdash \Delta}{\Gamma|\hat{\Psi}\mathcal{E} : A \vdash \Delta}\ (b)$$

2.

$$\frac{\Gamma \rhd H : A|\Delta \quad \Gamma|e : A \vdash \Delta}{(\hat{\Theta}e)[H] : (\Gamma \vdash \Delta)}$$

Proof: 1.(a) (resp. 1.(b)) is by a straightforward induction on e (resp. $\mathcal{E}$), using the first two rules in the left (resp. right) half of Figure 8. Statement 2. follows from (11) and the third rule on the left half of Figure 8. □

Corollary 3. *The following rule is admissible:*

$$\frac{\Gamma \rhd H : A|\Delta \quad \Gamma|\mathcal{E} : A \vdash \Delta}{\mathcal{E}[H] : (\Gamma \vdash \Delta)}$$

Proof: From 1.(b) and 2. in the previous lemma, together with $\hat{\Theta}\hat{\Psi}\mathcal{E} = \mathcal{E}$. □

Reduction for contexts. Let R be a reduction rule of $\underline{\lambda}\mu\mathsf{let}$. $\mathcal{E} \to_R \mathcal{E}'$ is defined by

$$\frac{S \to_R S'}{\mathsf{let}\, x = [\_]\, \mathsf{in}\, S \to_R \mathsf{let}\, x = [\_]\, \mathsf{in}\, S'}\ (i) \qquad \frac{N \to_R N'}{\mathcal{E}[[\_]N] \to_R \mathcal{E}[[\_]N']}\ (ii)$$
$$\frac{\mathcal{E} \to_R \mathcal{E}'}{\mathcal{E}[[\_]N] \to_R \mathcal{E}'[[\_]N]}\ (iii) \tag{17}$$

If $R = \eta_{\tilde{\mu}}$, we put in addition:

$$\frac{}{\mathsf{let}\, x = [\_]\, \mathsf{in}\, \mathcal{E}[\mathsf{h}(x)] \to_{\eta_{\tilde{\mu}}} \mathcal{E}}\ x \notin \mathcal{E} \tag{18}$$

In (17), (i) and (ii) express the principle that reduction in the $\underline{\lambda}\mu\mathsf{let}$ components of a contexts are reflected at the context level. Rule (iii) means that $\to_R$ at the level of contexts is compatible.

Lemma 2 (Isomorphism at the level co-terms/contexts). *Let R be a reduction rule of $\overline{\lambda}\mu\tilde{\mu}$ and R' the corresponding reduction rule of $\underline{\lambda}\mu\mathsf{let}$ according to Figure 11.*

1. *$e \to_R e'$ in $\overline{\lambda}\mu\tilde{\mu}$ iff $\hat{\Theta}e \to_{R'} \hat{\Theta}e'$ in Contexts.*
2. *$\mathcal{E} \to_{R'} \mathcal{E}'$ in Contexts iff $\hat{\Psi}\mathcal{E} \to_R \hat{\Psi}\mathcal{E}'$ in $\overline{\lambda}\mu\tilde{\mu}$.*

Proof: The "if" statements follow from the only if statements and the fact that $\hat{\Theta}$ and $\hat{\Psi}$ are inverse bijections. The "only if" statement 1. is proved by induction on $e \to_R e'$. There are four cases. First, $\eta_{\tilde{\mu}}$-reduction at root position; this corresponds to (18). Second, $e = \tilde{\mu}x.c$ and reduction in c; this corresponds to (i) in (17) and uses part 3(a) of Theorem 2. Third, $e = u :: e_0$ and reduction in u; this corresponds to (ii) in (17) and uses part 1(a) of Theorem 2. Fourth, $e = u :: e_0$ and reduction in e_0; this corresponds to (ii) in (17) and follows by IH. The "only if" statement 2. is proved by induction on $\mathcal{E} \to_R \mathcal{E}'$ in analogous fashion, using parts 1(b) and 3(b) of Theorem 2. □

Figure 23: Natural deduction semantics for $\overline{\lambda}\mu\tilde{\mu}$

$$\begin{array}{rcl} [\![x]\!] &=& x \\ [\![\lambda x.t]\!] &=& \lambda x.[\![t]\!] \\ [\![\mu a.c]\!] &=& \mu a.[\![c]\!] \end{array} \qquad [\![\langle t|e\rangle]\!] = [\![e]\!][\mathsf{h}([\![t]\!])] \qquad \begin{array}{rcl} [\![a]\!] &=& a([\_]) \\ [\![\tilde{\mu}x.c]\!] &=& \mathsf{let}\, x = [\_]\,\mathsf{in}\,[\![c]\!] \\ [\![u :: e]\!] &=& [\![e]\!][[\_][\![u]\!]] \end{array}$$

Corollary 4. *It holds that:*

$$\frac{\mathcal{E} \to_R \mathcal{E}'}{\mathcal{E}[H] \to_R \mathcal{E}'[H]}$$

Proof: Given $\mathcal{E} \to_R \mathcal{E}'$, we get $\hat{\Psi}\mathcal{E} \to_R \hat{\Psi}\mathcal{E}'$ in $\overline{\lambda}\mu\tilde{\mu}$ by the previous lemma. From 2(a) of Theorem 2, we get $\Theta(H, \hat{\Psi}\mathcal{E}) \to_R \Theta(H, \hat{\Psi}\mathcal{E}')$ in $\underline{\lambda}\mu\mathsf{let}$. From (11), this means $(\hat{\Theta}\hat{\Psi}\mathcal{E})[H] \to_R (\hat{\Theta}\hat{\Psi}\mathcal{E}')[H]$. By bijectivity, it follows $\mathcal{E}[H] \to_R \mathcal{E}'[H]$. □

6.2. Natural deduction semantics

We now formalize the semantics of $\overline{\lambda}\mu\tilde{\mu}$ as a morphism $[\![\_]\!] : \overline{\lambda}\mu\tilde{\mu} \to [\underline{\lambda}\mu\mathsf{let}{+}Contexts]$ comprising three mappings

- $[\![\_]\!] : \overline{\lambda}\mu\tilde{\mu} - Terms \longrightarrow \underline{\lambda}\mu\mathsf{let} - Terms$,
- $[\![\_]\!] : \overline{\lambda}\mu\tilde{\mu} - Commands \longrightarrow \underline{\lambda}\mu\mathsf{let} - Statements$,
- $[\![\_]\!] : \overline{\lambda}\mu\tilde{\mu} - CoTerms \longrightarrow \underline{\lambda}\mu\mathsf{let} - Contexts$,

defined by simultaneous recursion in Figure 23. This mapping is defined homomorphically on terms, like Θ; is defined on co-terms as $\hat{\Theta}$; and defines $[\![\langle t|e\rangle]\!]$ as the result of filling $\mathsf{h}([\![t]\!])$ in the hole of the context $[\![e]\!]$.

Recall that Θ maps a command $\langle t|e\rangle$ by calling $\Theta(\mathsf{h}(\Theta t), e)$, and that the latter transforms the successive left-introductions that appear at the top of the second argument into eliminations that accumulate on the first argument. This is the heart of the transformation of the sequent calculus format of $\overline{\lambda}\mu\tilde{\mu}$ to the natural deduction format of $\underline{\lambda}\mu\mathsf{let}$. However, this transformation of commands corresponds exactly to the idea of hole filling:

Proposition 3 (Semantics vs Θ). *1.* $[\![t]\!] = \Theta t$*; 2. (a)* $[\![e]\!] = \hat{\Theta}e$ *and (b)* $[\![e]\!][H] = \Theta(H, e)$*; 3.* $[\![c]\!] = \Theta c$*.*

Proof: By induction on t, e, and c. Observe that 2.(b) follows from 2.(a) and (11), and is used in the only interesting case of the proof, as follows. Let $c = \langle t|e\rangle$. Then:

$$\begin{array}{rcll} \Theta(\langle t|e\rangle) &=& \Theta(\mathsf{h}(\Theta t), e) & \text{(by def. } \Theta) \\ &=& [\![e]\!][\mathsf{h}(\Theta t)] & \text{(by IH 2.(b))} \\ &=& [\![e]\!][\mathsf{h}([\![t]\!])] & \text{(by IH 1.)} \\ &=& [\![\langle t|e\rangle]\!] & \text{(by def. } [\![\_]\!]) \end{array}$$

□

So, the semantics $[\![\_]\!]$ is the same mapping as Θ, at the level of terms and commands. In addition, we have argued that $\Theta(\_, e)$ is a "semantic context", which, we now see, corresponds to the context $[\![e]\!]$.

Proposition 4 (Soundness of semantics). *The following rules are admissible:*

1.
$$\frac{\Gamma \vdash t : A|\Delta}{\Gamma \vdash [\![t]\!] : A|\Delta} \qquad \frac{c : (\Gamma \vdash \Delta)}{[\![c]\!] : (\Gamma \vdash \Delta)} \qquad \frac{\Gamma|e : A \vdash \Delta}{\Gamma|[\![e]\!] : A \vdash \Delta}$$

2.
$$\frac{\Gamma \rhd H : A|\Delta \quad \Gamma|e : A \vdash \Delta}{[\![e]\!][H] : (\Gamma \vdash \Delta)}$$

Proof: 1. follows from part 1 (a) of Lemma 1, parts 1, 2 (a) and 3 of Proposition 3, and soundness of Θ. 2. follows from part 2 of Lemma 1 and part 2 (a) of Proposition 3. □

Theorem 5 (Semantics is isomorphism). *The semantics $[\![\_]\!]$ establishes an isomorphism at the levels of terms, commands/statements, and co-terms/contexts.*

Proof: At the level of terms, bijection is in Proposition 2, and isomorphism of reduction is part 1 of Theorem 2. At the level of commands/statements, bijection is in Proposition 2, and isomorphism of reduction is part 3 of Theorem 2. At the level of co-terms/contexts, bijection was remarked at (12), and isomorphism of reduction is by Lemma 2. □

6.3. Isomorphic point of view

The semantics $[\![\_]\!]$ comprises three isomorphisms, at the levels of terms, commands/statement, and co-terms/contexts. In particular, $[\![e]\!]$ is the context $\hat{\Theta}e$, a syntactic entity outside $\underline{\lambda}\mu\mathsf{let}$. For this reason, although the source of $[\![\_]\!]$ is $\overline{\lambda}\mu\tilde{\mu}$, its target is not $\underline{\lambda}\mu\mathsf{let}$, but $[\underline{\lambda}\mu\mathsf{let} + Contexts]$. In the latter, some typing and reduction structure is lifted to the level of contexts, whereas in $\overline{\lambda}\mu\tilde{\mu}$ the co-terms and corresponding structure are primitive.

This is an asymmetric situation. However, one easily feels it ought to be a partial view of the relationship between $\overline{\lambda}\mu\tilde{\mu}$ and $\underline{\lambda}\mu\mathsf{let}$, given the isomorphism proved in Theorem 2:

(i) Co-terms correspond to a (derived) syntactic class of $\underline{\lambda}\mu\mathsf{let}$. However, hole-expression do not correspond to any syntactic class in $\overline{\lambda}\mu\tilde{\mu}$.

(ii) $\hat{\Theta}e$ corresponds to the "semantic" context $\Theta(\_, e) : HoleExpressions \to Statements$. On the other hand, for each H, $\Psi(H, \_)$ is a function of type $CoTerms \to Commands$. What is the syntactic counterpart $\hat{\Psi}H$ of this "semantic" concept?

(iii) From a purely formal point of view, part 2 of Lemma 1 is incomplete, as it rephrases the third rule of the *left* half of Figure 8, but not of the right half. Similarly, Corollary 4 is incomplete, as it expresses part 2(a), but not part 2(b), of Theorem 2.

(iv) The semantics is an isomorphism. As such, its inverse should correspond to an inversion of point of view: $\underline{\lambda}\mu\mathsf{let}$ as the source language, $\overline{\lambda}\mu\tilde{\mu}$ as the semantics.

Figure 24: Isomorphic points of view

	$\overline{\lambda}\mu\tilde{\mu}$	$\underline{\lambda}\mu\mathsf{let}$
primitive	<u>co-terms e</u> - right associativity - left type - $\Gamma \vert e : A \vdash \Delta$	<u>hole-expressions H</u> - left associativity - right type - $\Gamma \rhd H : A \vert \Delta$
derived	<u>co-contexts $\mathcal{H}$</u> - command with a hole - hole on the right expects e - left associativity - right type - $\Gamma \rhd \mathcal{H} : A \vert \Delta$	<u>contexts $\mathcal{E}$</u> - statement with a hole - hole on the left expects H - right associativity - left type - $\Gamma \vert \mathcal{E} : A \vdash \Delta$

The complete, symmetric description is restored through the concept of *co-context*, a form of context (in the sense of expression with a hole) pertaining to $\overline{\lambda}\mu\tilde{\mu}$. As seen from Figure 24, this new concept quite evidently completes the picture. In $\underline{\lambda}\mu\mathsf{let}$, statements can be decomposed as $\mathcal{E}[H]$; in $\overline{\lambda}\mu\tilde{\mu}$, one can now decompose commands as $\mathcal{H}[e]$. So, in $\overline{\lambda}\mu\tilde{\mu}$, co-contexts are the "contexts" and co-terms are the "hole-expressions". In the same way that there is a bijection $\hat{\Theta}$, later promoted to an isomorphism, relating co-terms to contexts, there will be a bijection $\hat{\Psi}$ relating hole-expressions to co-contexts, later promoted to an isomorphism, when the typing and reduction structure of $\overline{\lambda}\mu\tilde{\mu}$ is lifted to co-contexts. Such system will be denoted $[\overline{\lambda}\mu\tilde{\mu} + CoContexts]$. Then, the semantics $[\![\_]\!]$ will be eventually extended to that system; and that system will be the range of a semantics of $\underline{\lambda}\mu\mathsf{let}$, contained in the inverse of $[\![\_]\!]$.[26]

6.4. Co-contexts vs hole-expressions

We first develop co-contexts as a meta-linguistic device of $\overline{\lambda}\mu\tilde{\mu}$, useful for describing commands and even reduction rules. Co-contexts $\mathcal{H}$ are commands of $\overline{\lambda}\mu\tilde{\mu}$ with a hole at the right end ($m \geq 0$):

$$\langle t | u_1 :: \cdots :: u_m :: [\_] \rangle$$

The hole of a co-context expects a co-term. The result of filling e in $\mathcal{H}$ is denoted $\mathcal{H}[e]$ and is a command. Describing a command c as $\mathcal{H}[e]$ is particularly useful when $e = a$ or $e = \tilde{\mu}x.c$, because then the part of c hidden at the right end is brought to surface.

Let *CoContexts* be the set of co-contexts. There is an algebra $\mathbb{H}$ with carrier *CoContexts* and the following operations: one constant $\langle t | [\_] \rangle$, for each t; and, for each u, a unary operation that maps $\mathcal{H}$ to $\mathcal{H}[u :: [\_]]$. In fact, each $\mathcal{H} \in CoContexts$ is generated by $\mathbb{H}$ in a unique way. In particular, co-contexts can be defined inductively as follows:

[26] In addition, referring to (iii) above, part 2 of Lemma 1 will be completed with part 2 of Lemma 3, and Corollary 4 will be completed with Corollary 6.

Figure 25: $\hat{\Theta} : CoContexts \to HoleExpressions$ and $\hat{\Psi} : HoleExpressions \to CoContexts$

$$\boxed{\begin{array}{rcl} \hat{\Theta}(\langle t|[\_]\rangle) & = & \mathsf{h}(\Theta t) \\ \hat{\Theta}(\mathcal{H}[u :: [\_]]) & = & (\hat{\Theta}\mathcal{H})\Theta u \end{array}} \qquad \boxed{\begin{array}{rcl} \hat{\Psi}(\mathsf{h}(M)) & = & \langle \Psi M|[\_]\rangle \\ \hat{\Psi}(HN) & = & (\hat{\Psi}H)[\Psi N :: [\_]] \end{array}}$$

$$\mathcal{H} := \langle t|[\_]\rangle \,|\, \mathcal{H}[u :: [\_]] \tag{19}$$

We define the mapping $\hat{\Psi}$ in the right half of Figure 25.[27] By a straightforward induction on H one proves that

$$(\hat{\Psi}H)[e] = \Psi(H, e) \ . \tag{20}$$

So, a command $\Psi(H, e)$ can be seen as $\mathcal{H}[e]$, with $\mathcal{H} = \hat{\Psi}H$.

Mapping $\hat{\Psi}$ on hole expressions has an inverse $\hat{\Theta}$ with domain on *CoContexts* and defined on the left half of Figure 25. Clearly:

$$\hat{\Psi}\hat{\Theta}\mathcal{H} = \mathcal{H} \qquad \hat{\Theta}\hat{\Psi}H = H \ . \tag{21}$$

6.5. η-reduction

In this subsection, we first present the reduction rule $\eta_\supset$ of $\underline{\lambda}\mu\mathsf{let}$. Next, we present the corresponding rule in $\overline{\lambda}\mu\tilde{\mu}$, which is best described with co-contexts.

We have defined in $\underline{\lambda}\mu\mathsf{let}$ the reduction rule $\eta_\supset$ as follows: $\lambda x.\mu a.a(Hx) \to \mu a.a(H)$, if $x, a \notin H$. At first sight, we may find it strange that the expression in the function position of the application to x is an H and not a term. But this formulation is more general and natural, given the organization of $\underline{\lambda}\mu\mathsf{let}$'s syntax: why would we restrict ourselves to the particular case $H = \mathsf{h}(M)$? In addition, such particular case would pose a problem: the contractum would be the η_μ-redex $\mu a.a(\mathsf{h}(M))$, and therefore, in the design of the $\eta_\supset$ rule, one would have to decide whether one η_μ-reduction step would be built in the $\eta_\supset$-rule or not. With the more general formulation, the design dilemma vanishes, and there is a separation of concerns: $\eta_\supset$ cares solely about the λ-abstraction, and η_μ is explicit called whenever needed, and not implicitly built in another rule.

Now, to such design corresponds in $\overline{\lambda}\mu\tilde{\mu}$ the $\eta_\supset$ rule we have announced in Figure 3: $\lambda x.\mu a.\mathcal{H}[x :: a] \to \mu a.\mathcal{H}[a]$, if $x, a \notin \mathcal{H}$ (the isomorphism with the $\eta_\supset$-rule of $\underline{\lambda}\mu\mathsf{let}$ is proved in Theorem 6 below). Informally, the $\eta_\supset$ rule of $\overline{\lambda}\mu\tilde{\mu}$ is

$$\lambda x.\mu a.\langle t|u_1 :: \cdots :: u_m :: x :: a\rangle \to \mu a.\langle t|u_1 :: \cdots :: u_m :: a\rangle \tag{22}$$

with $m \geq 0$. Its formal definition makes sense only now, after the development of the concept of co-context. In the version (22), one has to unfold the entire co-term of the redex $\lambda x.\mu a.\langle t|e\rangle$ to reveal $x :: a$, which is hidden at the bottom of the right associative structure e. The representation of $\langle t|e\rangle$ as $\mathcal{H}[x :: e]$ brings to light the hidden part.

[27] We re-use the names of the mappings in Figure 10. There is no danger since two mappings with the same name will have different domains.

The rule we propose for $\overline{\lambda}\mu\tilde{\mu}$ generalizes the η-rule that was proposed before in the literature [19]:

$$\lambda x.\mu b.\langle t|x :: a\rangle \to t \tag{23}$$

with $x, a \notin t$. Let $\mathcal{H} = \langle t|[\_]\rangle$. Then $x, a \notin \mathcal{H}$ and:

$$\begin{array}{rcl} \lambda x.\mu b.\langle t|x :: a\rangle & = & \lambda x.\mu a.\mathcal{H}[x :: b] \\ & \to_{\eta_\supset} & \mu b.\mathcal{H}[b] \\ & = & \mu b.\langle t|b\rangle \\ & \to_{\eta_\mu} & t \end{array}$$

The rule (23) corresponds to the particular case $H = \mathsf{h}(M)$ of the $\eta_\supset$-rule of $\underline{\lambda}\mu\mathsf{let}$ discussed above, and therefore shares with that particular case its design problems.

Theorem 6 (Isomorphism for $\eta_\supset$). *For terms:*

(a) $t \to_{\eta_\supset} t'$ in $\overline{\lambda}\mu\tilde{\mu}$ iff $\Theta t \to_{\eta_\supset} \Theta t'$ in $\underline{\lambda}\mu\mathsf{let}$.

(b) $M \to_{\eta_\supset} M'$ in $\underline{\lambda}\mu\mathsf{let}$ iff $\Psi M \to_{\eta_\supset} \Psi M'$ in $\overline{\lambda}\mu\tilde{\mu}$.

Similarly for co-terms/hole-expressions and commands/statements, as in Theorem 2.

Proof: In fact, this theorem is just the missing case $R = \eta_\supset = R'$ of Theorem 2. Accordingly, we just need to add to the proof of that result one base case to each induction.

For the first induction, which proves statements (a), we start with a remark:

$$H = \hat{\Theta}\mathcal{H} \Rightarrow \Theta(\mathcal{H}[e]) = \Theta(H, e) \ . \tag{24}$$

If $H = \hat{\Theta}\mathcal{H}$ then $\hat{\Psi}H = \mathcal{H}$. Hence, $\Theta(\mathcal{H}[e]) = \Theta((\hat{\Psi}H)[e]) \overset{(20)}{=} \Theta\Psi(H, e) = \Theta(H, e)$.

Let $t = \lambda x.\mu a.\mathcal{H}[x :: a] \to_{\eta_\supset} \mu a.\mathcal{H}[a] = t'$, with $x, a \notin \mathcal{H}$. Let $H = \hat{\Theta}\mathcal{H}$. Then $x, a \notin H$ and:

$$\begin{array}{rcll} \Theta t & = & \lambda x.\mu a.\Theta(\mathcal{H}[x :: a]) & \text{(by def. } \Theta) \\ & = & \lambda x.\mu a.\Theta(H, x :: a) & \text{(by (24))} \\ & = & \lambda x.\mu a.a(Hx) & \text{(by def. } \Theta) \\ & \to_{\eta_\supset} & \mu a.a(H) & \\ & = & \mu a.\Theta(H, a) & \text{(by def } \Theta) \\ & = & \mu a.\Theta(\mathcal{H}[a]) & \text{(by (24))} \\ & = & \Theta t' & \text{(by def } \Theta) \end{array}$$

For the second induction, which proves statements (b), let $M = \lambda x.\mu a.a(Hx) \to_{\eta_\supset} \mu a.a(H) = M'$, with $x, a \notin H$. Then $x, a \notin \hat{\Psi}H$ and:

$$\begin{array}{rcll} \Psi M & = & \lambda x.\mu a.\Psi(H, x :: a) & \text{(by def. } \Psi) \\ & = & \lambda x.\mu a.(\hat{\Psi}H)[x :: a] & \text{(by (20))} \\ & \to_{\eta_\supset} & \mu a.(\hat{\Psi}H)[a] & \\ & = & \mu a.\Psi(H, a) & \text{(by (20))} \\ & = & \Psi M' & \text{(by def. } \Psi) \end{array}$$

□

It is easy to see that $\eta_\supset$ on $\underline{\lambda}\mu\mathsf{let}$ enjoys subject reduction. This entails, with the help of the theorem above and soundness of Θ and Ψ, that $\eta_\supset$ on $\underline{\lambda}\mu\mathsf{let}$ enjoys subject reduction as well.

6.6. Co-contexts

We now equip the set *CoContexts* with typing and reduction rules. We adopt for co-contexts of $\overline{\lambda}\mu\tilde{\mu}$ the "intermediate" approach adopted before for contexts of $\underline{\lambda}\mu\mathsf{let}$ and explained in Subsection 6.1. Essentially, co-contexts inherit their typing and reduction rules from $\overline{\lambda}\mu\tilde{\mu}$, and so the "calculus of co-contexts" is defined only after the definition of $\overline{\lambda}\mu\tilde{\mu}$ is completed. We denote by

$$[\overline{\lambda}\mu\tilde{\mu} + CoContexts]$$

the system consisting of $\overline{\lambda}\mu\tilde{\mu}$ plus co-contexts equipped with typing and reduction rules.

Typing for co-contexts. The typing system for co-contexts of $\overline{\lambda}\mu\tilde{\mu}$ derives sequents

$$\Gamma \rhd \mathcal{H} : A|\Delta$$

Co-contexts are typed "on the right", with the right type being the type of the hole. This hole expects co-terms with that type - see Corollary 5.

The system has two typing rules:

$$\frac{\Gamma \vdash_{\overline{\lambda}\mu\tilde{\mu}} t : A|\Delta}{\Gamma \rhd \langle t|[\_]\rangle : A|\Delta}\ (i) \qquad \frac{\Gamma \rhd \mathcal{H} : A \supset B|\Delta \quad \Gamma \vdash_{\overline{\lambda}\mu\tilde{\mu}} u : A|\Delta}{\Gamma \rhd \mathcal{H}[u :: [\_]] : B|\Delta}\ (ii) \tag{25}$$

For instance, the rationale behind the second rules is as follows: if $\mathcal{H}$ is a command with a hole in the right end expecting a co-term of type $A \supset B$, and if in this hole we fill $u :: []$ with $u : A$, then the resulting expression is a command with a hole in its right end, now expecting a co-term of type B.

The sequents with tag $\overline{\lambda}\mu\tilde{\mu}$ in the rules (25) are not premisses, but derivability side conditions. In this, we follow the same style as in the typing system for contexts (16); also the typing system becomes coherent with the inductive definition (19) of co-contexts: it has an axiom and one one-premiss rule.

In the same way as the typing rules for contexts (16) have the sequent calculus format, the typing rules for co-contexts (25) have the natural deduction format: they operate on the r.h.s. of sequents, and (ii) is an elimination rule.

Lemma 3. *The following rules are admissible:*

1.
$$\frac{\Gamma \rhd \mathcal{H} : A \vdash \Delta}{\Gamma \rhd \hat{\Theta}\mathcal{H} : A \vdash \Delta}\ (a) \qquad \frac{\Gamma \rhd H : A \vdash \Delta}{\Gamma \rhd \hat{\Psi} H : A \vdash \Delta}\ (b)$$

2.
$$\frac{\Gamma \rhd H : A|\Delta \quad \Gamma|e : A \vdash \Delta}{(\hat{\Psi}H)[e] : (\Gamma \vdash \Delta)}$$

Proof: 1.(a) (resp. 1.(b)) is by a straightforward induction on $\mathcal{H}$ (resp. H). 2. follows from (20) and the third rule on the right half of Figure 8. $\square$

Corollary 5. *The following rule is admissible:*

$$\frac{\Gamma \rhd \mathcal{H} : A|\Delta \quad \Gamma|e : A \vdash \Delta}{\mathcal{H}[e] : (\Gamma \vdash \Delta)}$$

Proof: From 1.(a) and 2. in the previous proposition, together with $\hat{\Psi}\hat{\Theta}\mathcal{H} = \mathcal{H}$. □

Reduction for co-contexts. Let R be a reduction rule of $\overline{\lambda}\mu\tilde{\mu}$. $\mathcal{H} \to_R \mathcal{H}'$ is defined by:

$$\frac{t \to_R t'}{\langle t|[\_]\rangle \to_R \langle t'|[\_]\rangle}\ (i)$$
$$\frac{u \to_R u'}{\mathcal{H}[u :: [\_]] \to_R \mathcal{H}[u' :: [\_]]}\ (ii) \qquad \frac{\mathcal{H} \to_R \mathcal{H}'}{\mathcal{H}[u :: [\_]] \to_R \mathcal{H}'[u :: [\_]]}\ (iii) \tag{26}$$

Rules (i) and (ii) express that reduction in the $\overline{\lambda}\mu\tilde{\mu}$ components of a co-context is reflected at the co-context level. Rule (iii) ensures reduction at the level of co-contexts is compatible.

Lemma 4 (Isomorphism at the level co-contexts/hole-expressions). *Let R be a reduction rule of $\overline{\lambda}\mu\tilde{\mu}$ and R' the corresponding reduction rule of $\underline{\lambda}\mu\mathsf{let}$ according to the Figure 11.*

1. *$\mathcal{H} \to_R \mathcal{H}'$ in CoContexts iff $\hat{\Theta}\mathcal{H} \to_{R'} \hat{\Theta}\mathcal{H}'$ in $\underline{\lambda}\mu\mathsf{let}$.*
2. *$H \to_{R'} H'$ in $\underline{\lambda}\mu\mathsf{let}$ iff $\hat{\Psi}H \to_R \hat{\Psi}H'$ in CoContexts.*

Proof: The "if" statements follow from the only if statements and the fact that $\hat{\Theta}$ and $\hat{\Psi}$ are inverse bijections. The "only if" statement 1. is proved by induction on $\mathcal{H} \to_R \mathcal{H}'$. There are three cases. The first and second correspond to (i) and (ii) in (26) and use part 1(a) of Theorem 2. The third corresponds to (iii) in (17) and follows by IH. The "only if" statement 2. is proved by induction on $H \to_R H'$ in analogous fashion, using part 1(b) of Theorem 2. □

Corollary 6. *It holds that:*

$$\frac{\mathcal{H} \to_R \mathcal{H}'}{\mathcal{H}[e] \to_R \mathcal{H}'[e]}$$

Proof: Given $\mathcal{H} \to_R \mathcal{H}'$, we get $\hat{\Theta}\mathcal{H} \to_R \hat{\Theta}\mathcal{H}'$ in $\underline{\lambda}\mu\mathsf{let}$ by the previous lemma. From 2(b) of Theorem 2, we get $\Psi(\hat{\Theta}\mathcal{H}, e) \to_R \Psi(\hat{\Theta}\mathcal{H}', e)$ in $\underline{\lambda}\mu\mathsf{let}$. From (20), this means $(\hat{\Psi}\hat{\Theta}\mathcal{H})[e] \to_R (\hat{\Psi}\hat{\Theta}\mathcal{H}')[e]$. By bijectivity, it follows $\mathcal{H}[e] \to_R \mathcal{H}'[e]$. □

6.7. Epilogue

The natural deduction semantics of $\overline{\lambda}\mu\tilde{\mu}$ is extended to *CoContexts* in order to complete a mapping $[\![\_]\!] : [\overline{\lambda}\mu\tilde{\mu} + CoContexts] \to [\underline{\lambda}\mu\mathsf{let} + Contexts]$ comprising four components:

- $[\![\_]\!] : \overline{\lambda}\mu\tilde{\mu} - Terms \longrightarrow \underline{\lambda}\mu\mathsf{let} - Terms$,

Figure 26: Natural deduction semantics, completed

$$\begin{array}{rcl} [\![\langle t|[\_]\rangle]\!] & = & \mathsf{h}([\![t]\!]) \\ [\![\mathcal{H}[u :: [\_]]]\!] & = & [\![\mathcal{H}]\!][\![u]\!] \end{array}$$

- $[\![\_]\!] : \overline{\lambda}\mu\tilde{\mu} - Commands \longrightarrow \underline{\lambda}\mu\mathsf{let} - Statements$,
- $[\![\_]\!] : \overline{\lambda}\mu\tilde{\mu} - CoTerms \longrightarrow \underline{\lambda}\mu\mathsf{let} - Contexts$
- $[\![\_]\!] : \overline{\lambda}\mu\tilde{\mu} - CoContexts \longrightarrow \underline{\lambda}\mu\mathsf{let} - Hole - Expressions$

The last component is given in Figure 26, and maps co-contexts as $\hat{\Theta}$.

Proposition 5 (Semantics vs Θ, completed). *1. $[\![t]\!] = \Theta t$; 2. (a) $[\![e]\!] = \hat{\Theta} e$ and (b) $[\![e]\!][H] = \Theta(H, e)$; 3. $[\![c]\!] = \Theta c$; 4. $[\![\mathcal{H}]\!] = \hat{\Theta}\mathcal{H}$.*

Proof: Proposition 3 gives all but 4. The latter is by induction on $\mathcal{H}$, using 1. □

Proposition 6 (Soundness of semantics, completed). *The following rules are admissible:*

1.
$$\frac{\Gamma \vdash t : A|\Delta}{\Gamma \vdash [\![t]\!] : A|\Delta} \qquad \frac{c : (\Gamma \vdash \Delta)}{[\![c]\!] : (\Gamma \vdash \Delta)} \qquad \frac{\Gamma|e : A \vdash \Delta}{\Gamma|[\![e]\!] : A \vdash \Delta} \qquad \frac{\Gamma \rhd \mathcal{H} : A \vdash \Delta}{\Gamma \rhd [\![\mathcal{H}]\!] : A \vdash \Delta}$$

2.
$$\frac{\Gamma \rhd H : A|\Delta \quad \Gamma|e : A \vdash \Delta}{[\![e]\!][H] : (\Gamma \vdash \Delta)} \qquad \frac{\Gamma \rhd \mathcal{H} : A|\Delta \quad \Gamma|\mathcal{E} : A \vdash \Delta}{\mathcal{E}[[\![\mathcal{H}]\!]] : (\Gamma \vdash \Delta)}$$

Proof: Proposition 4 gives all but the last part of both 1. and 2. The last part of 1. follows from part 1 (a) of Lemma 3 and part 4 of Proposition 5. The last part of 2. follows from part 1 (a) of Lemma 3, part 4 of Proposition 5 and Corollary 3. □

Theorem 7 ($[\overline{\lambda}\mu\tilde{\mu} + CoContexts] \cong [\underline{\lambda}\mu\mathsf{let} + Contexts]$). *The semantics $[\![\_]\!]$ establishes an isomorphism at the levels of terms, commands/statements, co-terms/contexts, and co-contexts/hole-expressions.*

Proof: Given Theorem 5, it remains to argue the level of co-contexts/hole-expressions. Bijection was remarked in (21), and isomorphism of reduction is by Lemma 4. □

As a corollary, one can change the point of view and see $\overline{\lambda}\mu\tilde{\mu}$ as the semantics of $\underline{\lambda}\mu\mathsf{let}$. The semantics $[\![\_]\!]^{-1} : \underline{\lambda}\mu\mathsf{let} \to [\overline{\lambda}\mu\tilde{\mu} + CoContexts]$ comprises the three (by now well-known) isomorphisms:

- $[\![\_]\!]^{-1} : \underline{\lambda}\mu\mathsf{let} - Terms \longrightarrow \overline{\lambda}\mu\tilde{\mu} - Terms$,
- $[\![\_]\!]^{-1} : \underline{\lambda}\mu\mathsf{let} - Statements \longrightarrow \overline{\lambda}\mu\tilde{\mu} - Commands$,
- $[\![\_]\!]^{-1} : \underline{\lambda}\mu\mathsf{let} - Hole - Expressions \longrightarrow \overline{\lambda}\mu\tilde{\mu} - CoContexts.$

7. Conclusion

We conclude with some final remarks and comments on related and future work.

Final remarks. The results we proved about $\underline{\lambda}\mu\mathsf{let}$ are:

1. Subformula property (Theorem 1)
2. $\overline{\lambda}\mu\tilde{\mu} \cong \underline{\lambda}\mu\mathsf{let}$ (Theorems 2 and 6)
3. Strong normalisation (Corollary 1)
4. Subject reduction (Corollary 2)
5. CBN and CBV fragments (Theorem 3)
6. Connection with $\lambda\mu$ (Theorem 4)

The system $\underline{\lambda}\mu\mathsf{let}$ is not yet a fully-fledged natural deduction system because we have not considered other connectives, and therefore $\underline{\lambda}\mu\mathsf{let}$ is just a step "towards" a full system of natural deduction for classical logic. Still, as long as we are content with implication and a λ-calculus for classical logic, $\underline{\lambda}\mu\mathsf{let}$, by virtue of $\overline{\lambda}\mu\tilde{\mu} \cong \underline{\lambda}\mu\mathsf{let}$, is the canonical counterpart in natural deduction to the $\overline{\lambda}\mu\tilde{\mu}$ and its perfect account of classical logic symmetries.

Despite the isomorphism $\overline{\lambda}\mu\tilde{\mu} \cong \underline{\lambda}\mu\mathsf{let}$, the system $\underline{\lambda}\mu\mathsf{let}$ is not claimed to be as elegant as $\overline{\lambda}\mu\tilde{\mu}$; instead, $\underline{\lambda}\mu\mathsf{let}$ offers an alternative representation of $\overline{\lambda}\mu\tilde{\mu}$-term with inverse associativity of commands ("applicative terms"), with the right end at the surface. Therefore, in the same way as the $\overline{\lambda}\mu\tilde{\mu}$ representation is good for rules, like β, that act on the left end (dispensing with contexts), the $\underline{\lambda}\mu\mathsf{let}$ representation is preferable for rules, like $\eta_{\supset}$, which behave in the opposite way (dispensing with co-contexts).

In addition, the study of $\underline{\lambda}\mu\mathsf{let}$ already conducted in this paper proved to be rich and fruitful, as can be seen from the following high-level lessons:

1. The syntax of natural deduction is non-trivial. The issues are: the correct separation between what is primitive and what is derived; the correct organization into syntactic categories of the primitive syntax. Only then let-expressions can be added.
2. There is a correspondence (and a difference) between cut in sequent calculus and the substitution inference rule in natural deduction; the let-expressions constructor and the substitution inference rule are in Curry-Howard correspondence; let-expressions generalize delayed substitutions, and are not exclusive of CBV.
3. It is the reduction of let-expressions to the particular case of delayed substitutions that makes a natural deduction system CBN. In systems like $\lambda\mu$ (hence λ), such identification is implicitly made and even strengthened, through the treatment of substitution as a meta-operation.
4. The isomorphism $\Theta : \overline{\lambda}\mu\tilde{\mu} \to \underline{\lambda}\mu\mathsf{let}$ has several facets:
   (a) the proof-theoretical translation based on the idea of replacing each left-introduction inference by an elimination inference.
   (b) the mapping that inverts the associativity of "applicative terms".
   (c) the basis of a semantics that makes precise why co-terms and commands of $\overline{\lambda}\mu\tilde{\mu}$ are "contexts" and "hole-filling" instructions.
   (d) a formal recipe for the "read-back" into natural deduction; in particular, it is a recipe for the systematic generation of CBV λ-calculi in natural deduction style.

En passant, this paper contributes to $\overline{\lambda}\mu\tilde{\mu}$ the concept of co-contexts and the new $\eta_{\supset}$-reduction rule (in addition to the semantics and read-back into natural deduction).

Related work. In the body of the paper detailed comparison was made between the proposed system $\underline{\lambda}\mu\mathsf{let}$ and both $\lambda\mu$ and $\overline{\lambda}\mu\tilde{\mu}$ [25, 7], as well as between the CBV fragments of $\underline{\lambda}\mu\mathsf{let}$ and other proposals for CBV λ-calculi recently appeared in the literature [30, 19, 20].

In the recent studies about the correspondence between sequent calculus and natural deduction, one approach is to identify fragments of sequent calculus isomorphic to natural deduction. The initial result is $\lambda_H \cong \lambda$ of [9]. Other contributions of this kind are in [7, 21], although no isomorphism is claimed. The present paper belongs to another approach [36, 12], which pursues extensions of natural deduction isomorphic to full sequent calculus. The isomorphism $\overline{\lambda}\mu\tilde{\mu} \cong \underline{\lambda}\mu\mathsf{let}$ extends to classical logic the intuitionistic $\lambda^{\mathsf{Gtz}} \cong \lambda_{\mathsf{Nat}}$ of [12].

Dyckhoff and Lengrand [8] prove an equational correspondence [32] between LJQ (the Q subsystem of intuitionistic sequent calculus) and Moggi's computational λ-calculus [22]. An isomorphism is to be expected between LJQ and the intuitionistic, Q subsystem of $\underline{\lambda}\mu\mathsf{let}$.

Moggi [22] explained the difference between substitution and let-expressions as the difference between the *composition principles* of two different but related categories: some category with a monad and the corresponding Kleisli category. In the present paper we explain that cut and let are composition principles of two different but related proof-systems: sequent calculus and natural deduction.

Future work. $\underline{\lambda}\mu\mathsf{let}$ can be a starting point for continuing the study of natural deduction for classical logic. One questions is the already mentioned extension to other logical connectives. Another concerns proof search in natural deduction. In the same way as, for proof reduction, there are situations where the sequent calculus format of $\overline{\lambda}\mu\tilde{\mu}$ has advantages over the natural deduction format of $\underline{\lambda}\mu\mathsf{let}$ and vice versa, the isomorphism $\overline{\lambda}\mu\tilde{\mu} \cong \underline{\lambda}\mu\mathsf{let}$ does not guarantee that conducting proof search in one system or the other is the same thing. It remains to see the relative advantages of searching in one system vs the other.

This paper produces, through a proof-theoretical analysis, new calculi for the future investigation of CBV. It is now a different task to put those tools to work. In spite of differing from other CBV λ-calculi in the literature, the new calculus $\underline{\lambda}\mu\mathsf{let}_{\mathsf{Q}}$, being isomorphic to $\overline{\lambda}\mu\tilde{\mu}_{\mathsf{Q}}$, validates the usual cps semantics [7]. The difference may be telling, however, if we consider reductions instead of equations, and operational aspects like standardization and abstract machines.

Acknowledgments: The author is supported by FCT, through Centro de Matemática, Universidade do Minho. Paul Taylor's macros were used for typesetting diagrams. The author thanks the anonymous referees for detailed comments.

References

[1] Zena M. Ariola, Hugo Herbelin, and Amr Sabry. A proof-theoretic foundation of abortive continuations. *Higher-Order and Symbolic Computation*, 20(4):403–429, 2007.

[2] Kensuke Baba, Sachio Hirokawa, and Ken-etsu Fujita. Parallel reduction in type free lambda/mu-calculus. *Electr. Notes Theor. Comput. Sci.*, 42:52–66, 2001.

[3] H.P. Barendregt. *The Lambda Calculus*. North-Holland, 1984.

[4] R Bloo and H. Geuvers. Explicit substitution: on the edge of strong normalization. *Theoretical Computer Science*, 211(1-2):375–395, 1999.

[5] B. Boričić. On sequence-conclusion natural deduction systems. *Journal of Philosofical Logic*, 14:359–377, 1985.

[6] I. Cervesato and F. Pfenning. A linear spine calculus. *Journal of Logic and Computation*, 13(5):639–688, 2003.

[7] P.-L. Curien and H. Herbelin. The duality of computation. In *Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming (ICFP '00), Montreal, Canada, September 18-21, 2000*, SIGPLAN Notices 35(9), pages 233–243. ACM, 2000.

[8] R. Dyckhoff and S. Lengrand. Call-by-value lambda calculus and LJQ. *Journal of Logic and Computation*, 17:1109–1134, 2007.

[9] J. Espírito Santo. Revisiting the correspondence between cut-elimination and normalisation. In *Proceedings of ICALP'2000*, volume 1853 of *Lecture Notes in Computer Science*, pages 600–611. Springer-Verlag, 2000.

[10] J. Espírito Santo. An isomorphism between a fragment of sequent calculus and an extension of natural deduction. In M. Baaz and A. Voronkov, editors, *Proceedings of LPAR'02*, volume 2514 of *Lecture Notes in Artificial Intelligence*, pages 354–366. Springer-Verlag, 2002.

[11] J. Espírito Santo. Delayed substitutions. In Franz Baader, editor, *Proceedings of RTA'07*, volume 4533 of *Lecture Notes in Computer Science*, pages 169–183. Springer-Verlag, 2007.

[12] J. Espírito Santo. The λ-calculus and the unity of structural proof theory. *Theory of Computing Systems*, 45:963–994, 2009.

[13] J. Espírito Santo. Towards a canonical classical natural deduction system. In A. Dawar and H. Veith, editors, *Proceedings of CSL'10*, volume 6247 of *Lecture Notes in Computer Science*, pages 290–304. Springer-Verlag, 2010.

[14] M. Felleisen, D. Friedman, E. Kohlbecker, and B. Duba. Reasoning with continuations. In *1st Symposium on Logic and Computer Science*. IEEE, 1986.

[15] G. Gentzen. Investigations into logical deduction. In M. E. Szabo, editor, *The collected papers of Gerhard Gentzen*, pages 68–131. North Holland, 1969.

[16] J-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*. Cambridge University Press, 1989.

[17] T. Griffin. A formulae-as-types notion of control. In *ACM Conf. Principles of Programming Languages*. ACM Press, 1990.

[18] H. Herbelin. A λ-calculus structure isomorphic to a Gentzen-style sequent calculus structure. In L. Pacholski and J. Tiuryn, editors, *Proceedings of CSL'94*, volume 933 of *Lecture Notes in Computer Science*, pages 61–75. Springer-Verlag, 1995.

[19] H. Herbelin. C'est maintenant qu'on calcule, 2005. Habilitation Thesis.

[20] H. Herbelin and S. Zimmermann. An operational account of call-by-value minimal and classical lambda-calculus in "natural deduction" form. In *Proceedings of Typed Lambda Calculi and Applications'09*, volume 5608 of *Lecture Notes in Computer Science*, pages 142–156. Springer-Verlag, 2009.

[21] K. Kikuchi. Call-by-name reduction and cut-elimination in classical logic. *Annals of Pure and Applied Logic*, 153:38–65, 2008.

[22] E. Moggi. Computational lambda-calculus and monads. Technical Report ECS-LFCS-88-86, University of Edinburgh, 1988.

[23] S. Negri and J. von Plato. *Structural Proof Theory*. Cambridge University Press, 2001.

[24] C-H.L. Ong and C.A. Stewart. A Curry-Howard foundation for functional computation with control. In *Proc. of Symposium on Principles of Programming Languages (POPL'97)*, pages 215–217. ACM Press, 1997.

[25] M. Parigot. $\lambda\mu$-calculus: an algorithmic interpretation of classic natural deduction. In *Int. Conf. Logic Prog. Automated Reasoning*, volume 624 of *Lecture Notes in Computer Science*, pages 190–201. Springer Verlag, 1992.

[26] G. Plotkin. Call-by-name, call-by-value and the λ-calculus. *Theoretical Computer Science*, 1:125–159, 1975.

[27] E. Polonovski. Strong normalization of lambda-mu-mu-tilde with explicit substitutions. In Igor (Ed.) Walukiewicz, editor, *Proc. of 7th Int. Conference on Foundations of Software Sciences and Computation Structures (FoSSaCS 2004)*, volume 2987 of *Lecture Notes in Computer Science*, pages 423–437. Springer-Verlag, 2004.

Figure 27: CBV fragments of $\overline{\lambda}\mu\tilde{\mu}$

$\overline{\lambda}\mu\tilde{\mu}_{\mathsf{Q}}$			
t, u	$::=$	$W \,\vert\, \mu a.c$	
W	$::=$	$x \,\vert\, \lambda x.t$	
e	$::=$	$a \,\vert\, W :: e \,\vert\, \tilde{\mu}x.c$	
c	$::=$	$\langle t \vert e\rangle$	
$(\beta_{\supset})$	$\langle \lambda x.t \vert W :: e\rangle$	$\rightarrow$	$\langle W/x\rangle\langle t \vert e\rangle$
$(\sigma_{\tilde{\mu}v})$	$\langle W/x\rangle c$	$\rightarrow$	$[W/x]c$
(σ_{μ})	$\langle e/a\rangle c$	$\rightarrow$	$[e/a]c$
(η_{μ})	$\mu a.\langle t \vert a\rangle$	$\rightarrow$	t, if $a \notin t$
$(\eta_{\tilde{\mu}})$	$\tilde{\mu}x.\langle x \vert e\rangle$	$\rightarrow$	e, if $x \notin e$

$\overline{\lambda}\tilde{\mu}$			
W	$::=$	$x \,\vert\, \lambda(x, a).c$	
e	$::=$	$a \,\vert\, W :: e \,\vert\, \tilde{\mu}x.c$	
c	$::=$	$\langle W \vert e\rangle$	
$(\beta_{\supset})$	$\langle \lambda(x, a).c \vert W :: e\rangle$	$\rightarrow$	$[e/a][W/x]c$
$(\sigma_{\tilde{\mu}v})$	$\langle W/x\rangle c$	$\rightarrow$	$[W/x]c$
$(\eta_{\tilde{\mu}})$	$\tilde{\mu}x.\langle x \vert e\rangle$	$\rightarrow$	e, if $x \notin e$

[28] D. Prawitz. *Natural Deduction. A Proof-Theoretical Study.* Almquist and Wiksell, Stockholm, 1965.

[29] N. Rehof and M. Sorensen. The λ_{Δ}-calculus. In *TACS'94*, volume 789 of *Lecture Notes in Computer Science*. Springer Verlag, 1994.

[30] J. Rocheteau. $\lambda\mu$-calculus and duality: call-by-name and call-by-value. In J. Giesl (Ed.), editor, *Proc. of RTA 2005*, volume 3467 of *Lecture Notes in Computer Science*, pages 204–218. Springer-Verlag, 2005.

[31] K. Rose. Explicit substitutions: Tutorial & survey. Technical Report LS-96-3, BRICS, 1996.

[32] A. Sabry and M. Felleisen. Reasoning about programms in continuation-passing-style. *LISP and Symbolic Computation*, 6(3/4):289–360, 1993.

[33] A. Sabry and P. Wadler. A reflection on call-by-value. *ACM Transactions on Programming Languages and Systems*, 19(6):916–941, 1997.

[34] G. Stalmarck. Normalization theorems for full first order classical natural deduction. *The Journal of Symbolic Logic*, 56(1):129–149, 1991.

[35] A. Troelstra and H. Schwitchtenberg. *Basic Proof Theory.* Cambridge University Press, second edition, 2000.

[36] J. von Plato. Natural deduction with general elimination rules. *Arquive for Mathematical Logic*, 40(7):541–567, 2001.

A. Fragments of $\overline{\lambda}\mu\tilde{\mu}$

For reader's convenience, we recall the fragments of $\overline{\lambda}\mu\tilde{\mu}$ defined in [7].

The CBV fragments of $\overline{\lambda}\mu\tilde{\mu}$ are defined in Fig. 27.

$\overline{\lambda}\mu\tilde{\mu}_{\mathsf{Q}}$ is obtained from $\overline{\lambda}\mu\tilde{\mu}$ through two restrictions: (i) $u :: e$ is constrained to $W :: e$, where W is a value; (ii) σ is constrained to redexes of the form $\langle W/x\rangle c$, in order to avoid the critical pair with π.

Figure 28: CBN fragments of $\overline{\lambda}\mu\tilde{\mu}$

$\overline{\lambda}\mu\tilde{\mu}_{\mathsf{T}}$			
	t, u	$::=$	$x \mid \lambda x.t \mid \mu a.c$
	e	$::=$	$E \mid \tilde{\mu}x.c$
	E	$::=$	$a \mid u :: E$
	c	$::=$	$\langle t \vert e \rangle$
$(\beta_{\supset})$	$\langle \lambda x.t \vert u :: E \rangle$	$\rightarrow$	$\langle u/x \rangle \langle t \vert E \rangle$
$(\sigma_{\tilde{\mu}})$	$\langle u/x \rangle c$	$\rightarrow$	$[u/x]c$
$(\sigma_{\mu n})$	$\langle E/a \rangle c$	$\rightarrow$	$[E/a]c$
(η_{μ})	$\mu a.\langle t \vert a \rangle$	$\rightarrow$	t, if $a \notin t$

$\overline{\lambda}\mu$			
	t, u	$::=$	$x \mid \lambda x.t \mid \mu a.c$
	E	$::=$	$a \mid u :: E$
	c	$::=$	$\langle t \vert E \rangle$
$(\beta_{\supset})$	$\langle \lambda x.t \vert u :: E \rangle$	$\rightarrow$	$\langle [u/x]t \vert E \rangle$
$(\sigma_{\mu n})$	$\langle E/a \rangle c$	$\rightarrow$	$[E/a]c$
(η_{μ})	$\mu a.\langle t \vert a \rangle$	$\rightarrow$	t, if $a \notin t$

$\overline{\lambda}\tilde{\mu}$ is obtained from $\overline{\lambda}\mu\tilde{\mu}_{\mathsf{Q}}$ after two steps. First, μ-abstraction is removed from the syntax. This is achieved by restricting commands to the form $\langle W|e\rangle$ and hiding μ-abstraction occurring under a λ through a "double abstraction" $\lambda(x,a).S$. After the first step, the class of terms can be dispensed with, and values are generated by the grammar $W ::= x \mid \lambda(x,a).S \mid \lambda x.W$. The second step is to drop values of the form $\lambda x.W$, having in mind that these can be seen as $\lambda(x,a).\langle W|a\rangle$, a fresh. The double abstraction makes the β rule generate two substitutions, which is expected since β in $\overline{\lambda}\tilde{\mu}$ somehow amalgamates β and π of $\overline{\lambda}\mu\tilde{\mu}_{\mathsf{Q}}$. Since μ-abstraction was removed from the syntax, rule η_{μ} is dropped.

The CBN fragments of $\overline{\lambda}\mu\tilde{\mu}$ are defined in Fig. 28.

$\overline{\lambda}\mu\tilde{\mu}_{\mathsf{T}}$ is obtained from $\overline{\lambda}\mu\tilde{\mu}$ through two restrictions: (i) every $u :: e$ is such that e is a co-value, that is, a co-term not of the form $\tilde{\mu}x.c$. This entails that a co-value has the restricted form of an "applicative context" E - a co-term in the class $E ::= a|u :: E$. Moreover, $\tilde{\mu}$-abstraction in $\overline{\lambda}\mu\tilde{\mu}_{\mathsf{T}}$ can only occur as the right component of commands; (ii) π is constrained to redexes of the form $\langle E/a\rangle c$, in order to avoid the critical pair with σ.

Rule $\eta_{\tilde{\mu}}$ can be dropped in $\overline{\lambda}\mu\tilde{\mu}_{\mathsf{T}}$, given the presence of rule σ, and if we are only interested in observing reduction at the level of terms or statements. This is so because an $\eta_{\tilde{\mu}}$-reduction at the level of commands is also a σ-reduction: $\langle t|\tilde{\mu}x.\langle x|e\rangle\rangle \rightarrow_{\sigma} \langle t|e\rangle$.

$\overline{\lambda}\mu$ is obtained from $\overline{\lambda}\mu\tilde{\mu}_{\mathsf{T}}$ by removing $\tilde{\mu}$-abstraction from the syntax. This is equivalent to requiring commands to have the form $\langle t|E\rangle$. Since $\tilde{\mu}$-abstraction was removed from the syntax, rules σ is dropped, and rule β has to compensate the absence of σ by executing immediately the substitution it generates.