A completeness result for $\lambda\mu$

Phillipe Audebaud[‡] Steffen van Bakel[§]

Inria Sophia Antipolis, 2004 route des Lucioles, BP 93, 06902 Sophia Antipolis, France
Phillipe.Audebaud@ens-lyon.fr,svb@doc.ic.ac.uk

Abstract

We study the expressivity of Parigot's $\lambda\mu$-calculus, and show that each statement $\Gamma \vdash_{\text{LK}} \Delta$ that is provable in Gentzen's LK has a proof in $\lambda\mu$. This result is obtained through defining an interpretation from *nets* from the $\mathcal{X}$-calculus into both the λ-calculus and $\lambda\mu$; X enjoys the full Curry-Howard isomorphism for (the implicative fragment of) LK, and cut-elimination in LK is fully represented by reduction in $\mathcal{X}$.
This interpretation will be shown to preserve reduction in $\mathcal{X}$ via equality in the target calculi, and to preserve typeability using the standard double negation translation of types. Using the fact that, in $\lambda\mu$, we can inhabit $\neg\neg A \rightarrow A$ for all types A, it is then shown that if $P : \Gamma \vdash_{\mathcal{X}} \Delta$ for the X-net P, then the translation of this proof into $\lambda\mu$ gives a valid derivation, and provides a witness for the sequent.

1 Introduction

The sequent calculus LK, introduced by Gentzen [6], is a logical system in which the rules only introduce connectives (but on both sides of a sequent), on the contrary to natural deduction which uses introduction and elimination rules. The only way to eliminate a connective is to eliminate the whole formula in which it appears, with an application of the (cut)-rule. Gentzen's calculus for classical logic LK allows sequents of the form $A_1,\ldots,A_n \vdash B_1,\ldots,B_m$, where $A_1,\ldots,A_n$ is to be understood as $A_1 \wedge \ldots \wedge A_n$ and $B_1,\ldots,B_m$ is to be understood as $B_1 \vee \ldots \vee B_m$. Thus, LK appears as a very symmetrical system.

For this calculus, a *cut-elimination procedure* has been defined that eliminates all applications of the (cut)-rule from the proof of a sequent, generating a proof in *normal form* of the same sequent, that is, with no cut. It is defined via local rewriting steps, reductions of the proof-tree, which has the flavour of the evaluation of explicit substitutions [4], now a wide area of interest in programming theory.

The calculus $\mathcal{X}$, as presented in [1, 10] represents a correspondence *à la* Curry-Howard-de Bruijn for LK$(\rightarrow)$, the implicational fragment of LK, bringing together the various features of two diffeent approaches: that of Urban [14] and that of Curien and Herbelin [5]. The aim of this paper is to relate $\mathcal{X}$ to the standard presentation of $\lambda\mu$, as we target the analysis of two subreduction systems, both designed to avoid unrecoverable critical pairs. We will show that there exist faithful mappings from $\mathcal{X}$ to both the λ-calculus and the $\lambda\mu$-calculus, which establishes a strong link between provable sequents in LK$(\rightarrow)$ and the $\lambda\mu$-calculus.

The relevance of this result can be understood from observing that the $\mathcal{X}$-calculus is symmetric for LK, while the $\lambda\mu$-calculus is not. However, we will show that the latter is expressive

[‡] École Normale Supérieure de Lyon, 46 Allée d'Italie 69364 Lyon 07, FRANCE

[§] On sabbatical leave from Department of Computing, Imperial College London, 180 Queen's Gate London SW7 2BZ, U.K.

enough to reflect the propagation rules of $\mathcal{X}$, exhibiting that the actual loss is due to natural deduction presentation of the calculus, which forces for the choice of a distinguished conclusion. The symmetric nature of the $\mathcal{X}$-calculus reveals itself from the fact that reduction is not confluencent; as the target calculus is confluent, the price to pay is, as for the pure λ-calculus case, to restrict ourselves to particular subsystems of the full reduction, that do not cause unrecoverable critical pairs to occur.

From the logical point of view, the natural deduction presentation introduces a lack of symmetry which is going to require more work for the translation of $\mathcal{X}$-terms into $\lambda\mu$]-terms. Our translation therefore will consist of first applying a CPS-like transformation, followed by a recovery of the type information for the global derivation tree:

$$\mathcal{X} \;\rightarrow\; \lambda\mu_{\mathrm{CPS}} \;\rightarrow\; \lambda\mu$$

This will allow us to prove:

Theorem *If $\Gamma \vdash_{\mathrm{LK}} \Delta$ in* LK$(\rightarrow)$*, then there exists a type T, $\lambda\mu$-term M, and contexts Γ',Δ' such that $\Gamma' \vdash_{\lambda\mu} M_P : T \mid \Delta'$ such that Γ,Δ can be obtained from Γ',Δ' by erasure of names.*

This result is obtained via the interpretation of $\mathcal{X}$ circuits into pure λ-terms, which requires double-negation translation on types and loses syntactical distinction between inputs and outputs. As we deal with a fragment of classical logic, a minimum requirement is to extend the pure λ-calculus is this direction. So λ_C-Calculus (where C stands for Griffin's C operator) [7, 9] or Parigot's $\lambda\mu$-Calculus [12] come in mind. Since we want sockets and plugs being kept distinct from each other, we favour the second solution.

Limiting ourselves to the implicative fragment of LK might seem to be too much of a restriction, but this not so. In fact, extending the calculus with (rules and constructs for) the logical connectives $\wedge,\vee,\forall,\exists,\neg$ is straightforward, and brings no added complexity for achievable results. Also, arrow types are the natural types for the λ-calculus and $\lambda\mu$.

2 The $\mathcal{X}$-calculus

2.1 From LK to a calculus

As mentioned in the introduction, $\mathcal{X}$ is inspired by the sequent calculus, so it is worthwhile to recall some of the principles.

Definition 2.1 (LK$(\rightarrow)$) The sequent calculus we consider has only implication, no structural rules and a changed axiom. It offers an extremely natural presentation of the classical propositional calculus with implication, and is a variant of system LK.

It has four rules: *axiom*, *right introduction* of the arrow, *left introduction* and *cut*.

$$(ax):\ \frac{}{\Gamma,A \vdash A,\Delta} \qquad (cut):\ \frac{\Gamma \vdash A,\Delta \quad \Gamma,A \vdash \Delta}{\Gamma \vdash \Delta}$$

$$({\Rightarrow}R):\ \frac{\Gamma,A \vdash B,\Delta}{\Gamma \vdash A{\Rightarrow}B,\Delta} \qquad ({\Rightarrow}L):\ \frac{\Gamma \vdash A,\Delta \quad \Gamma,B \vdash \Delta}{\Gamma,A{\Rightarrow}B \vdash \Delta}$$

As one knows, the rule (cut) plays a major role in proofs, since for proof theoreticians, cut-free proofs enjoy nice properties; proof reductions by cut-elimination have been proposed by Gentzen. Those reductions become the fundamental principle of computation in $\mathcal{X}$.

The Curry-Howard correspondence for $\mathcal{X}$ with classical propositional calculus is achieved by giving propositions names; those that appear in the left part of a sequent receive names like $x,y,z,\ldots$ and those that appear in the right part of a sequent receive name like $\alpha,\beta,\gamma,\ldots$, and to associate formulae with types.

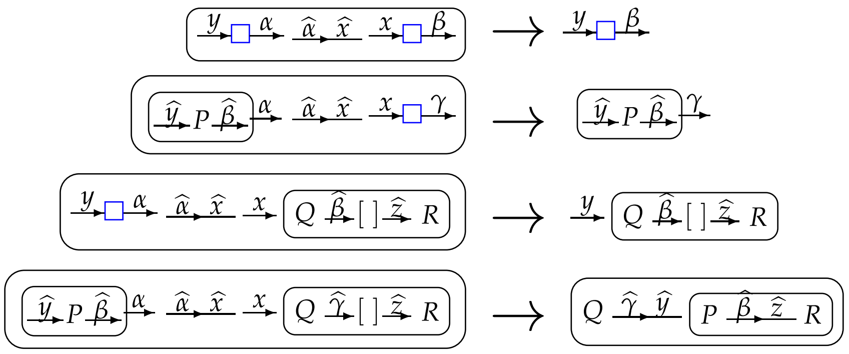


Figure 1: Diagrammatical representation of the logical rules

2.2 Syntax

The circuits that are the objects of $\mathcal{X}$ are built with three kinds of building stones, or constructors, called *capsule*, *export* and *import*. In addition there is an operator we call *cut*, which is handy for describing circuit construction, and which will be eliminated eventually by *rules*. These four will be the natural representatives for the four logical rules given above.

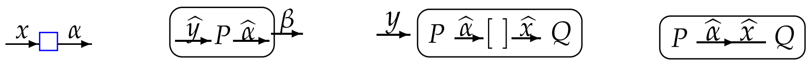
Circuits are connected through *wires* that are named. In our description wires are oriented. This means we know in which direction the 'ether running through our circuits' moves, and can say when a wire provides an entrance to a circuit or when a wire provides an exit. Thus we make the distinction between exit wires which we call *plugs* and enter wires which we call *sockets*. Plugs are named with Greek letters $\alpha,\beta,\gamma,\delta,\ldots$ and sockets are named with Latin letters $x,y,z,\ldots$.

When connecting two circuits P and Q, we may suppose that P has a plug α and Q has a socket x which we want to connect together to create a flow from P to Q. After the link has been established, the wires have been plugged, and the name of the plug and the name of the socket are forgotten. To be more precise, in $P\widehat{\alpha} \dagger \widehat{x}Q$, the name α is bound over P and the name x is bound over Q, bound in the interaction. We use the *"hat"*-notation, keeping in line with the old tradition of *Principia Mathematica* [16], writing $\hat{x}$ to say that x is bound.

Definition 2.2 (Syntax) The circuits of the $\mathcal{X}$-calculus are defined by the following grammar, where $x,y,\ldots$ range over the infinite set of *sockets*, and α,β over the infinite set of *plugs*.

$$\begin{array}{rcccccccc} P,Q & ::= & \langle x{\cdot}\alpha\rangle & \mid & \widehat{y}P\widehat{\beta}{\cdot}\alpha & \mid & P\widehat{\beta}[y]\widehat{x}Q & \mid & P\widehat{\alpha}\dagger\widehat{x}Q \\ & & \textit{capsule} & & \textit{export} & & \textit{import} & & \textit{cut} \end{array}$$

Diagrammatically, we represent the basic circuits as:



Notice that, using the intuition sketched above, for example, the connector β is supposed not to occur outside of P; this is formalised below by Definition 2.3 and Barendregt's Convention (see also below).

The calculus, defined by the reduction rules (Section 2.3) explains in detail how cuts are distributed through circuits to be eventually erased at the level of capsules.

We spoke above about bound names; we will introduce now formally those notions with that of free sockets and plugs into $\mathcal{X}$.

Definition 2.3 The *free sockets* and *free plugs* in a circuit are:

$$\begin{array}{llllll}
fs(\langle x{\cdot}\alpha\rangle) & = & \{x\} & fp(\langle x{\cdot}\alpha\rangle) & = & \{\alpha\} \\
fs(\widehat{x}P\widehat{\beta}{\cdot}\alpha) & = & fs(P)\setminus\{x\} & fp(\widehat{x}P\widehat{\beta}{\cdot}\alpha) & = & (fp(P)\setminus\{\beta\})\cup\{\alpha\} \\
fs(P\widehat{\alpha}\,[y]\,\widehat{x}Q) & = & fs(P)\cup\{y\}\cup(fs(Q)\setminus\{x\}) & fp(P\widehat{\alpha}\,[y]\,\widehat{x}Q) & = & (fp(P)\setminus\{\alpha\})\cup fp(Q) \\
fs(P\widehat{\alpha}\dagger\widehat{x}Q) & = & fs(P)\cup(fs(Q)\setminus\{x\}) & fp(P\widehat{\alpha}\dagger\widehat{x}Q) & = & (fp(P)\setminus\{\alpha\})\cup fp(Q)
\end{array}$$

A socket x or plug α which is not free is called *bound*, written $x \in bs(P)$ and $\alpha \in bp(P)$. We will write $x \notin fs(P,Q)$ for $x \notin fs(P)$ & $x \notin fs(Q)$.

We will normally adopt Barendregt's convention (called convention on variables by Barendregt, but here it will be a convention on names).

Convention on names. In a term or in a statement, a name is never both bound *and* free in the same context.

As the main concept is that of a name, we define only renaming, i.e., substitution of a name by another name as it makes no sense to define the substitution of a name by a term. The definition of renaming relies on Barendregt's convention on names; if a binding, say $\hat{x}P$ of x in P violates Barendregt's convention, one can get it back by renaming, i.e., $\hat{y}P[y/x]$; this renaming can be internalised (see [2]).

2.3 The rules

For reduction, it is important to know when a socket or a plug is introduced, i.e. is connectable, i.e. is exposed and unique. Informally, a circuit P introduces a socket x if P is constructed from subcircuits which do not contain x as free socket, so x only occurs at the "top level." This means that P is either a mediator with a middle connector $[x]$ or a capsule with left part x. Similarly, a circuit introduces a plug α if it is an export that "creates" α or a capsule with right part α (Urban [14] uses the terminology "freshly introduce").

Definition 2.4 (Introduction [1]) $(P$ *introduces* $x)$: $P = \langle x{\cdot}\beta\rangle$ or $P = R\widehat{\alpha}\,[x]\,\widehat{y}Q$, with $x \notin fs(R,Q)$.
$(P$ *introduces* $\alpha)$: $P = \langle y{\cdot}\alpha\rangle$ or $P = \widehat{x}Q\widehat{\beta}{\cdot}\alpha$ with $\alpha \notin fp(Q)$.

We first present a simple family of reduction rules. They say how to reduce a circuit that cuts subcircuits that both introduce connectors.

Definition 2.5 (Logical Reduction [1]) The logical rules are (assume that the terms of the left-hand sides of the rules *introduce* the socket x and the plug α)

$$\begin{array}{rrcl}
(cap): & \langle y{\cdot}\alpha\rangle\widehat{\alpha}\dagger\widehat{x}\langle x{\cdot}\beta\rangle & \to & \langle y{\cdot}\beta\rangle \\
(exp): & (\widehat{y}P\widehat{\beta}{\cdot}\alpha)\widehat{\alpha}\dagger\widehat{x}\langle x{\cdot}\gamma\rangle & \to & \widehat{y}P\widehat{\beta}{\cdot}\gamma \\
(med): & \langle y{\cdot}\alpha\rangle\widehat{\alpha}\dagger\widehat{x}(P\widehat{\beta}\,[x]\,\widehat{z}Q) & \to & P\widehat{\beta}\,[y]\,\widehat{z}Q \\
(exp\text{-}imp): & (\widehat{y}P\widehat{\beta}{\cdot}\alpha)\widehat{\alpha}\dagger\widehat{x}(Q\widehat{\gamma}\,[x]\,\widehat{z}R) & \to & \left\{\begin{array}{l}(Q\widehat{\gamma}\dagger\widehat{y}P)\widehat{\beta}\dagger\widehat{z}R \\ Q\widehat{\gamma}\dagger\widehat{y}(P\widehat{\beta}\dagger\widehat{z}R)\end{array}\right.
\end{array}$$

Their diagrammatical representation is given in Figure 1.

Notice that, in rule $(exp\text{-}imp)$, in addition to the conditions for introduction of the connectors that are active in the cut ($\alpha \notin fp(P)$ and $x \notin fs(Q,R)$) we can also state that $\beta \notin fp(Q)\backslash\{\gamma\}$, as well as that $y \notin fs(R)\backslash\{z\}$, due to Barendregt's convention.

We now need to define how to reduce a cut circuit in case when one of its sub-circuits does not introduce a socket or a plug. This requires to extend the syntax with two new operators that we call *activated* cuts:

$$P \ ::= \ \ldots \mid P\widehat{\alpha} \not\dagger \widehat{x}Q \mid P\widehat{\alpha} \backslash\!\!\!\dagger \widehat{x}Q$$

Reduction on terms with activated cuts will make sure these are propagated through the terms.

Definition 2.6 (Activating the cuts [1])

$$\begin{array}{rlcll}
(a\not\dagger): & P\widehat{\alpha} \dagger \widehat{x}Q & \to & P\widehat{\alpha} \not\dagger \widehat{x}Q, & \textit{if } P \textit{ does not introduce } \alpha \\
(\backslash\!\!\!\dagger a): & P\widehat{\alpha} \dagger \widehat{x}Q & \to & P\widehat{\alpha} \backslash\!\!\!\dagger \widehat{x}Q, & \textit{if } Q \textit{ does not introduce } x
\end{array}$$

Notice that both side-conditions can be valid simultaneously, thereby validating both rewrite rules at the same moment. This gives, in fact, a *critical pair* or *superposition* for our notion of reduction, and is the cause for the loss of confluence.

We will now define how to propagate a activated cut through sub-circuits. The direction of the activating shows in which direction the cut should be propagated, hence the two sets of reduction rules.

Definition 2.7 (Propagation Reduction [1]) The rules of propagation are:

Left propagation

$$\begin{array}{rrcl}
(d\not\dagger): & \langle y{\cdot}\alpha\rangle\widehat{\alpha} \not\dagger \widehat{x}P & \to & \langle y{\cdot}\alpha\rangle\widehat{\alpha} \dagger \widehat{x}P \\
(cap\not\dagger): & \langle y{\cdot}\beta\rangle\widehat{\alpha} \not\dagger \widehat{x}P & \to & \langle y{\cdot}\beta\rangle, \quad \beta \neq \alpha \\
(exp\text{-}out\not\dagger): & (\widehat{y}Q\widehat{\beta}{\cdot}\alpha)\widehat{\alpha} \not\dagger \widehat{x}P & \to & (\widehat{y}(Q\widehat{\alpha} \not\dagger \widehat{x}P)\widehat{\beta}{\cdot}\gamma)\widehat{\gamma} \dagger \widehat{x}P, \gamma \textit{ fresh} \\
(exp\text{-}in\not\dagger): & (\widehat{y}Q\widehat{\beta}{\cdot}\gamma)\widehat{\alpha} \not\dagger \widehat{x}P & \to & \widehat{y}(Q\widehat{\alpha} \not\dagger \widehat{x}P)\widehat{\beta}{\cdot}\gamma, \gamma \neq \alpha \\
(imp\not\dagger): & (Q\widehat{\beta}[z]\widehat{y}R)\widehat{\alpha} \not\dagger \widehat{x}P & \to & (Q\widehat{\alpha} \not\dagger \widehat{x}P)\widehat{\beta}\,[z]\,\widehat{y}(R\widehat{\alpha} \not\dagger \widehat{x}P) \\
(cut\not\dagger): & (Q\widehat{\beta} \dagger \widehat{y}R)\widehat{\alpha} \not\dagger \widehat{x}P & \to & (Q\widehat{\alpha} \not\dagger \widehat{x}P)\widehat{\beta} \dagger \widehat{y}(R\widehat{\alpha} \not\dagger \widehat{x}P)
\end{array}$$

Right propagation

$$\begin{array}{rlcll}
(\backslash\!\!\!\dagger d): & P\widehat{\alpha} \backslash\!\!\!\dagger \widehat{x}\langle x{\cdot}\beta\rangle & \to & P\widehat{\alpha} \dagger \widehat{x}\langle x{\cdot}\beta\rangle & \\
(\backslash\!\!\!\dagger cap): & P\widehat{\alpha} \backslash\!\!\!\dagger \widehat{x}\langle y{\cdot}\beta\rangle & \to & \langle y{\cdot}\beta\rangle, & y \neq x \\
(\backslash\!\!\!\dagger exp): & P\widehat{\alpha} \backslash\!\!\!\dagger \widehat{x}(\widehat{y}Q\widehat{\beta}{\cdot}\gamma) & \to & \widehat{y}(P\widehat{\alpha} \backslash\!\!\!\dagger \widehat{x}Q)\widehat{\beta}{\cdot}\gamma & \\
(\backslash\!\!\!\dagger imp\text{-}out): & P\widehat{\alpha} \backslash\!\!\!\dagger \widehat{x}(Q\widehat{\beta}[x]\widehat{y}R) & \to & P\widehat{\alpha} \dagger \widehat{z}((P\widehat{\alpha} \backslash\!\!\!\dagger \widehat{x}Q)\widehat{\beta}\,[z]\,\widehat{y}(P\widehat{\alpha} \backslash\!\!\!\dagger \widehat{x}R)), & z \textit{ fresh} \\
(\backslash\!\!\!\dagger imp\text{-}in): & P\widehat{\alpha} \backslash\!\!\!\dagger \widehat{x}(Q\widehat{\beta}[z]\widehat{y}R) & \to & (P\widehat{\alpha} \backslash\!\!\!\dagger \widehat{x}Q)\widehat{\beta}\,[z]\,\widehat{y}(P\widehat{\alpha} \backslash\!\!\!\dagger \widehat{x}R), z \neq x & \\
(\backslash\!\!\!\dagger cut): & P\widehat{\alpha} \backslash\!\!\!\dagger \widehat{x}(Q\widehat{\beta} \dagger \widehat{y}R) & \to & (P\widehat{\alpha} \backslash\!\!\!\dagger \widehat{x}Q)\widehat{\beta} \dagger \widehat{y}(P\widehat{\alpha} \backslash\!\!\!\dagger \widehat{x}R) &
\end{array}$$

The rules $(exp\text{-}out\not\dagger)$ and $(\backslash\!\!\!\dagger imp\text{-}out)$ deserve some attention. For instance, in the left-hand side of $(exp\text{-}out\not\dagger)$, α is not introduced, hence α occurs more than once in $\widehat{y}Q\widehat{\beta}{\cdot}\alpha$, that is once after the dot and again in Q. The occurrence after the dot is dealt with separately by creating a new name γ. Note that the cut associated with that γ is then unactivated; this is because, after the cut has been pushed through $\widehat{y}(Q\widehat{\alpha} \not\dagger \widehat{x}P)\widehat{\beta}{\cdot}\gamma$ (so leaves a circuit with no activated cut), the resulting term $(\widehat{y}R\widehat{\beta}{\cdot}\gamma)\widehat{\gamma} \dagger \widehat{x}P$ needs to be considered in its entirety: although we now that now γ is introduced, we know not if x is. A similar reasoning holds for x in $(\backslash\!\!\!\dagger imp\text{-}out)$ and a new name z is created and the external cut is not active.

2.4 Call-by-name and call-by-value

In this section we will define two sub-systems of reduction, that correspond to call-by-name (cbn) and call-by-value (cbv) reduction. Notice that this is essentially different from the approach of [15], where, as in $\overline{\lambda}\mu\tilde{\mu}$, only one notion of reduction is defined.

As mentioned above, when P does not introduce α and Q does not introduce x, $P\widehat{\alpha} \dagger \widehat{x}Q$ is a *superposition*, meaning that two rules, namely $(a\not\dagger)$ and $(\backslash\!\!\!\dagger a)$, can both be fired.

The *critical pair* $\langle P\widehat{\alpha} \not\nearrow \widehat{x}Q, P\widehat{\alpha} \nwarrow\!\!\!\setminus \widehat{x}Q\rangle$. may lead to different irreducible terms. This is to say that the reduction relation $\rightarrow$ is *not confluent*. Non-determinism is a a key feature of both classical logic and rewriting logic.

We introduce two strategies which explicitly favour one kind of activating whenever the above critical pair occurs. Consider a term $P\widehat{\alpha} \dagger \widehat{x}Q$ where P does not introduce α and Q does not introduce x, intuitively CBV tends to push Q through P and CBN tends to do the other way around.

Definition 2.8 • The CBV strategy only activates a cut via $(a\not\nearrow)$ when it could be activated in two ways; we write $P \rightarrow_{\mathrm{V}} Q$ in that case. We can reformulate this as the reduction system obtained by replacing rule $(\nwarrow\!\!\!\setminus a)$ by:

$$\begin{array}{rcll}(\nwarrow\!\!\!\setminus a): & P\widehat{\alpha} \dagger \widehat{x}Q & \rightarrow & P\widehat{\alpha} \nwarrow\!\!\!\setminus \widehat{x}Q, \ \textit{if } P \textit{ introduces } \alpha \\ & & & \textit{and } Q \textit{ does not introduce } x.\end{array}$$

- The CBN strategy only activates such a cut via $(\nwarrow\!\!\!\setminus a)$; like above, we write $P \rightarrow_{\mathrm{N}} Q$. Likewise, we can reformulate this as the reduction system obtained by replacing rule $(a\not\nearrow)$ by:

$$\begin{array}{rcll}(a\not\nearrow): & P\widehat{\alpha} \dagger \widehat{x}Q & \rightarrow & P\widehat{\alpha} \not\nearrow \widehat{x}Q, \ \textit{if } Q \textit{ introduces } x \\ & & & \textit{and } P \textit{ does not introduce } \alpha.\end{array}$$

3 Typing for $\mathcal{X}$

We will now formally define a notion of type assignment on $\mathcal{X}$, which will establish the Curry-Howard-de Bruijn isomorphism between $\mathcal{X}$ and LK$(\rightarrow)$.

Definition 3.1 (Types and Contexts) *i*) The set of types is defined by the grammar:

$$A, B \ ::= \ \varphi \mid A \rightarrow B.$$

The types considered in this paper are normally known as *simple* (or *Curry*) types.

ii) A *context of sockets* Γ is a mapping from sockets to types, denoted as a finite set of *statements* $x : A$, such that the *subject* of the statements (x) are distinct. We write $\Gamma, x : A$ for the context defined by:

$$\begin{array}{rcll}\Gamma, x : A & = & \Gamma \cup \{x : A\}, & \text{if } \Gamma \text{ is not defined on } x \\ & = & \Gamma, & \text{otherwise}\end{array}$$

So, when writing a context as $\Gamma, x : A$, this implies that $x : A \in \Gamma$, or Γ is not defined on x. When we write Γ_1, Γ_2 we mean the union of Γ_1 and Γ_2 when Γ_1 and Γ_2 are coherent (if Γ_1 contains $x : A_1$ and Γ_2 contains $x : A_2$ then $A_1 = A_2$).

iii) Contexts of *plugs* Δ are defined in a similar way.

Definition 3.2 (Typing for $\mathcal{X}$) *i*) *Type judgements* are expressed via a ternary relation $P : \Gamma \vdash \Delta$, where Γ is a context of *sockets* and Δ is a context of *plugs*, and P is a circuit. We say that P is the *witness* of this judgement.

ii) *Type assignment for* $\mathcal{X}$ is defined by the following sequent calculus:

$$(cap): \ \frac{}{\langle y{\cdot}\alpha\rangle : \Gamma, y{:}A \vdash \alpha{:}A, \Delta} \qquad (imp): \ \frac{P : \Gamma \vdash \alpha{:}A, \Delta \quad Q : \Gamma, x{:}B \vdash \Delta}{P\widehat{\alpha}\,[y]\,\widehat{x}Q : \Gamma, y{:}A{\rightarrow}B \vdash \Delta}$$

$$(exp): \ \frac{P : \Gamma, x{:}A \vdash \alpha{:}B, \Delta}{\widehat{x}P\widehat{\alpha}{\cdot}\beta : \Gamma \vdash \beta{:}A{\rightarrow}B, \Delta} \qquad (cut): \ \frac{P : \Gamma \vdash \alpha{:}A, \Delta \quad Q : \Gamma, x{:}A \vdash \Delta}{P\widehat{\alpha} \dagger \widehat{x}Q : \Gamma \vdash \Delta}$$

We write $P : \Gamma \vdash \Delta$ if there exists a derivation that has this judgement in the bottom line.

Γ and Δ carry the types of the free connectors in P, as unordered sets. There is no notion of type for P itself, instead the derivable statement shows how P is connectable.

The Curry-Howard property for the implicative fragment of LK is easily achieved by erasing all term-information.

The soundness result of simple type assignment with respect to reduction is stated as usual:

Theorem 3.3 (Witness reduction [1]) *If $P : \Gamma \vdash \Delta$, and $P \to Q$, then $Q : \Gamma \vdash \Delta$.*

4 Interpreting the λ-calculus

The expressive power of $\mathcal{X}$ is illustrated in [1] by showing that the λ-calculus [3], $\lambda\mathbf{x}$, $\lambda\mu$, and $\overline{\lambda}\mu\tilde{\mu}$ can be faithfully interpreted. Using the notion of Curry type assignment, assignable types are preserved by the interpretation.

In part, the interpretation results are detailed and precise, and deal with explicit substitution as well. In fact, the interpretation encompasses CBV and CBN reduction. However, in $\mathcal{X}$ we have no need of two separate interpretation functions, but will define only *one*. Combining this with the two sub-reduction systems $\to_{\text{V}}$ and $\to_{\text{N}}$ we can encode the the CBV- and CBN-λ-calculus.

We assume the reader to be familiar with the λ-calculus [3]; we just recall the definition of lambda terms and β-contraction.

Definition 4.1 (Lambda terms and reduction [3])

i) The set Λ of *lambda terms* is defined by the syntax:

$$M \ ::= \ x \mid \lambda x.M \mid M_1M_2$$

ii) The reduction relation $\to_\beta \subseteq \Lambda \times \Lambda$ is defined as the contextual, reflexive, symmetric, and transitive (i.e. compatible [3]) closure of the rule:

$$(\lambda x.M)N \ \to_\beta \ M[N/x]$$

iii) The notion of reduction $\to_\beta$ can be restricted to *Call by Value* reduction by: the set of *values* $\subseteq \Lambda$ is defined by the syntax:

$$V \ ::= \ x \mid \lambda x.M$$

Then the *Call by Value* reduction relation $\to_{\text{V}}$ is defined as the compatible closure of the rule:

$$(\lambda x.M)V \ \to_\beta \ M[V/x]$$

The full reduction system is then called *Call by Name*, and we will write $\to_{\text{N}}$ when necessary.

This calculus has a notion of type assignment that corresponds nicely to implicative propositional logic, in the framework of natural deduction.

Definition 4.2 (Type assignment for the λ-calculus)

$$(Ax): \ \frac{}{\Gamma, x{:}A \vdash_\lambda x : A} \qquad (\to I): \ \frac{\Gamma, x{:}A \vdash_\lambda M : B}{\Gamma \vdash_\lambda \lambda x.M : A\to B}$$

$$(\to E): \ \frac{\Gamma \vdash_\lambda M : A\to B \qquad \Gamma \vdash_\lambda N : A}{\Gamma \vdash_\lambda MN : B}$$

The direct encoding of the λ-calculus into $\mathcal{X}$ is defined by:

Definition 4.3 (Interpreting Λ in $\mathcal{X}$ [1])

$$\begin{array}{rcl} \ulcorner x \lrcorner^{\lambda}_{\alpha} & = & \langle x{\cdot}\alpha \rangle \\ \ulcorner \lambda x.M \lrcorner^{\lambda}_{\alpha} & = & \widehat{x} \ulcorner M \lrcorner^{\lambda}_{\beta} \widehat{\beta} \cdot \alpha \\ \ulcorner MN \lrcorner^{\lambda}_{\alpha} & = & \ulcorner M \lrcorner^{\lambda}_{\gamma} \widehat{\gamma} \dagger \widehat{x} (\ulcorner N \lrcorner^{\lambda}_{\beta} \widehat{\beta} \; [x] \; \widehat{y} \langle y{\cdot}\alpha \rangle) \end{array}$$

Observe that every sub-term of $\ulcorner M \lrcorner^{\lambda}_{\alpha}$ has exactly one free plug. It is worthwhile to notice that the interpretation function $\ulcorner \cdot \lrcorner^{\lambda}_{\alpha}$ does not generate a confluent sub-calculus. We illustrate this by the following:

Example 4.4 ([1]) We have

$$\ulcorner (\lambda x.xx)(yy) \lrcorner^{\lambda}_{\alpha} \;\to\; (\langle y{\cdot}\sigma \rangle \widehat{\sigma} \; [y] \; \widehat{z} \langle z{\cdot}\gamma \rangle) \widehat{\gamma} \dagger \widehat{x} (\langle x{\cdot}\tau \rangle \widehat{\tau} \; [x] \; \widehat{u} \langle u{\cdot}\alpha \rangle)$$

This circuit now has one cut only, that can be activated in two ways (notice that neither γ nor x is introduced here). This reduces to both:

$$\langle y{\cdot}\sigma \rangle \widehat{\sigma} \; [y] \; \widehat{z} (\langle z{\cdot}\tau \rangle \widehat{\tau} \; [z] \; \widehat{u} \langle u{\cdot}\alpha \rangle)$$
and
$$\langle y{\cdot}\sigma \rangle \widehat{\sigma} \; [y] \; \widehat{z} ((\langle y{\cdot}\sigma \rangle \widehat{\sigma} \; [y] \; \widehat{z} \langle z{\cdot}\tau \rangle) \widehat{\tau} \; [z] \; \widehat{u} \langle u{\cdot}\alpha \rangle)$$

Notice that both reductions return normal forms, and that these are different.

[1] shows the following result.

Theorem 4.5 (Simulation of cbn and cbv [1])

i) If $M \to_{\text{N}} N$ then $\ulcorner M \lrcorner^{\lambda}_{\gamma} \to_{\text{N}} \ulcorner N \lrcorner^{\lambda}_{\gamma}$.

ii) If $M \to_{\text{V}} N$ then $\ulcorner M \lrcorner^{\lambda}_{\gamma} \to_{\text{V}} \ulcorner N \lrcorner^{\lambda}_{\gamma}$.

Using these two results, the significance of Example 4.4 becomes clearer. Notice that

$$\begin{array}{rcl} \ulcorner (\lambda x.xx)(yy) \lrcorner^{\lambda}_{\alpha} & \to_{\text{V}} & \langle y{\cdot}\sigma \rangle \widehat{\sigma} \; [y] \; \widehat{z} (\langle z{\cdot}\tau \rangle \widehat{\tau} \; [z] \; \widehat{u} \langle u{\cdot}\alpha \rangle) \\ \ulcorner (\lambda x.xx)(yy) \lrcorner^{\lambda}_{\alpha} & \to_{\text{N}} & \langle y{\cdot}\sigma \rangle \widehat{\sigma} \; [y] \; \widehat{z} ((\langle y{\cdot}\sigma \rangle \widehat{\sigma} \; [y] \; \widehat{z} \langle z{\cdot}\tau \rangle) \widehat{\tau} \; [z] \; \widehat{u} \langle u{\cdot}\alpha \rangle) \end{array}$$

In the λ-calculus, $(\lambda x.xx)(yy)$ has different normal forms with respect to cbv and cbn λ-reduction (respectively $(\lambda x.xx)(yy)$ and $yy(yy)$), which are *both* interpreted in $\mathcal{X}$. Both, different from the λ-calculus, the term $\ulcorner (\lambda x.xx)(yy) \lrcorner^{\lambda}_{\alpha}$ in $\mathcal{X}$ is *not* a normal form for $\to_{\text{V}}$; it contains cuts. But, true to its nature, the cbv-reduction will not return $\ulcorner yy(yy) \lrcorner^{\lambda}_{\alpha}$, but, instead, returns the term $\ulcorner yy \lrcorner$ with the duplication $\ulcorner zz \lrcorner$ 'waiting to be applied' in the continuation.

[1] also shows that typeability is preserved by $\ulcorner \cdot \lrcorner^{\lambda}_{\alpha}$:

Theorem 4.6 ([1]) *If $\Gamma \vdash_{\lambda} M : A$, then $\ulcorner M \lrcorner^{\lambda}_{\alpha} : \Gamma \vdash \alpha : A$.*

5 The $\lambda\mu$-calculus

Parigot [12] presented the $\lambda\mu$-calculus as a calculus which extends the proofs-as-programs paradigm to classical logic. The $\lambda\mu$-calculus gives a natural deduction system, which allows to deal with multi conclusions by choosing at most one *active* formula at once. This is achieved by introducing two kinds of variables, as found in more recent calculus like $\mathcal{X}$ itself, $\overline{\lambda}\mu\tilde{\mu}$ [5], and others.

Definition 5.1 ($\lambda\mu$ terms) Terms of the $\lambda\mu$–calculus are generated by the grammar

$$\begin{array}{rrcl} (\textit{terms}): & M,N & ::= & x \mid \lambda x.M \mid MN \mid \mu\alpha.C \\ (\textit{commands}): & C & ::= & [\alpha]M \end{array}$$

where x ranges over (ordinary) term variables, and α over formula names (also called μ-variables).

The set of *values* is defined by:

$$(\mathit{values}):\quad V \;::=\; x \mid \lambda x.M \mid \mu\alpha.[\beta]V$$

We emphasise the clear distinction made above between regular terms and the so called commands; we use this separation for convenience only as our interpretations rely on it. The original presentation of the calculus would have the case $\mu\alpha.[\beta]M$ in the syntax for terms.

Definition 5.2 ($\lambda\mu$ REDUCTION) Reduction is defined as the compatible closure of the following reduction rules:

$$\begin{array}{rrcl}
\mathit{logical}(\beta): & \lambda x.M\,N & \to & M\langle N/x\rangle \\
\mathit{structural}(\mu): & (\mu\alpha.C)\,M & \to & \mu\alpha.C[[\alpha]\Box M/[\alpha]\Box] \\
\mathit{renaming}(\nu): & [\beta]\mu\alpha.C & \to & C\langle\alpha/\beta\rangle \\
\mathit{erasing}: & \mu\alpha.[\alpha]M & \to & M \text{ \textit{if} } \alpha \text{ \textit{does not occur in} } M.
\end{array}$$

As usual, the substitution mechanism $C[[\alpha]\Box M/[\alpha]\Box]$ used in the μ-reduction rule consists of replacing *recursively* every occurrence in the command C of a command $[\alpha]N$ *labelled* with α, by the command $[\alpha]N\,M$.

Call-by-Value reduction is defined by restricting the rules as follows:

$$\begin{array}{rcl}
\lambda x.M\,V & \to & M\langle V/x\rangle \\
(\mu\alpha.C)\,V & \to & \mu\alpha.C[[\alpha]\Box V/[\alpha]\Box]
\end{array}$$

Equipped with these reductions, the $\lambda\mu$-calculus is well known to be confluent.

In this paper, strictly speaking, we do not need to deal with μ-reduction, but, instead, with the compounded reduction defined by:

$$(\nu\circ\mu):\quad [\alpha](\mu\beta.C)\,(\lambda x.M) \;\to\; C\langle\lambda x.M\cdot\alpha/\beta\rangle$$

where the substitution mechanism consists of replacing recursively in C every occurrence of a command of the shape $[\beta]N$ by the command $[\alpha]N\,\lambda x.M$.

In this paper, we shall assign types to $\lambda\mu$-terms much along the same lines as for the λ-calculus. Actually, we will use more general judgements such as $\Gamma \vdash_{\lambda\mu} M:T \mid \Delta$ where Δ holds types for μ-variables, and is void as far as pure λ-calculus-terms are concerned. Formally:

Definition 5.3 (TYPE ASSIGNMENT FOR $\lambda\mu$) Type assignment for $\lambda\mu$ is defined by the following natural deduction system:

$$\begin{array}{rc}
(Ax): & \dfrac{}{\Gamma,x{:}A \vdash x:A \mid \Delta} \\[2ex]
(\to I): & \dfrac{\Gamma,x{:}A \vdash M:B \mid \Delta}{\Gamma \vdash \lambda x.M : A{\to}B \mid \Delta} \\[2ex]
(\to E): & \dfrac{\Gamma \vdash M:A{\to}B \mid \Delta \qquad\qquad \Gamma \vdash N:A \mid \Delta}{\Gamma \vdash MN:B \mid \Delta} \\[2ex]
(\mu): & \dfrac{\Gamma \vdash C:\textsc{Cmd} \mid \alpha{:}A,\Gamma}{\Gamma \vdash \mu\alpha.C:A \mid \Gamma} \\[2ex]
(\textsc{Cmd}): & \dfrac{\Gamma \vdash M:A \mid \alpha{:}A,\Gamma}{\Gamma \vdash [\alpha]M:\textsc{Cmd} \mid \alpha{:}A,\Gamma}
\end{array}$$

6 Interpreting $\mathcal{X}$ into the λ-calculus

As can be expected, the interpretation of $\mathcal{X}$'s circuits into pure λ-terms requires a double-negation translation on types and loses syntactical distinction between inputs and outputs. As we deal with a fragment of classical logic, a minimum requirement is to extend the pure λ-calculus is this direction.

However, in this section we will show that we can still faithfully interpret $\mathcal{X}$ into the λ-calculus, and obtain a type-preservation result using the 'double negation' technique. A similar result was obtained in [10]; the main difference between that result and the one obtained here is that we interpret left- and right-cuts differently.

6.1 Call by Name

We will now show that we can interpret the CBN-subreduction system of $\mathcal{X}$ in the CBN-λ-calculus. It should be noted that, in the CBN reduction system, a left-cut $P\widehat{\alpha} \not\nearrow \widehat{x}Q$ is only generated if Q introduces x, so if $Q = \langle x{\cdot}\beta\rangle$, or $Q = R\widehat{\beta}\,[x]\,\widehat{y}S$, and y not free in R,S; this observation will prove important when dealing with activation and deactivation rules.

The CBN-interpretation of circuits in $\mathcal{X}$ as terms in Λ is defined as follows:

Definition 6.1 (CBN INTERPRETATION)

$$\begin{array}{rcl}
\llbracket \langle x{\cdot}\alpha\rangle \rrbracket^{\lambda}_{\mathrm{N}} & \triangleq & x\lambda u.\alpha u \\
\llbracket \widehat{x}P\widehat{\alpha}\cdot\beta \rrbracket^{\lambda}_{\mathrm{N}} & \triangleq & \beta\lambda x\alpha.\,\llbracket P \rrbracket^{\lambda}_{\mathrm{N}} \\
\llbracket P\widehat{\alpha}\,[y]\,\widehat{x}Q \rrbracket^{\lambda}_{\mathrm{N}} & \triangleq & y\lambda u.(\lambda x.\,\llbracket Q \rrbracket^{\lambda}_{\mathrm{N}})(u\lambda\alpha.\,\llbracket P \rrbracket^{\lambda}_{\mathrm{N}}) \\
\llbracket P\widehat{\alpha} \dagger \widehat{x}Q \rrbracket^{\lambda}_{\mathrm{N}} = \llbracket P\widehat{\alpha} \not\nwarrow \widehat{x}Q \rrbracket^{\lambda}_{\mathrm{N}} & \triangleq & (\lambda x.\,\llbracket Q \rrbracket^{\lambda}_{\mathrm{N}})(\lambda\alpha.\,\llbracket P \rrbracket^{\lambda}_{\mathrm{N}}) \\
\llbracket P\widehat{\alpha} \not\nearrow \widehat{x}\langle x{\cdot}\beta\rangle \rrbracket^{\lambda}_{\mathrm{N}} & \triangleq & (\lambda\alpha.\,\llbracket P \rrbracket^{\lambda}_{\mathrm{N}})(\lambda x.\beta x) \\
\llbracket P\widehat{\alpha} \not\nearrow \widehat{x}(Q\widehat{\beta}\,[x]\,\widehat{y}R) \rrbracket^{\lambda}_{\mathrm{N}} & \triangleq & (\lambda\alpha.\,\llbracket P \rrbracket^{\lambda}_{\mathrm{N}})(\lambda x.(\lambda y.\,\llbracket R \rrbracket^{\lambda}_{\mathrm{N}})(x\lambda\beta.\,\llbracket Q \rrbracket^{\lambda}_{\mathrm{N}}))
\end{array}$$

Notice that, defining

$$\begin{array}{rcl}
\llbracket \langle x{\cdot}\beta\rangle \rrbracket_{\mathrm{A}} & = & \beta x \\
\llbracket Q\widehat{\beta}\,[x]\,\widehat{y}R \rrbracket_{\mathrm{A}} & = & (\lambda y.\,\llbracket R \rrbracket^{\lambda}_{\mathrm{N}})(x\lambda\beta.\,\llbracket Q \rrbracket^{\lambda}_{\mathrm{N}})
\end{array}$$

for the Q that appears in $P\widehat{\alpha} \not\nearrow \widehat{x}Q$, we have

$$\llbracket Q \rrbracket^{\lambda}_{\mathrm{N}} \;=\; x\lambda u.\,\llbracket Q \rrbracket_{\mathrm{A}}[u/x]$$

and could have defined

$$\llbracket P\widehat{\alpha} \not\nearrow \widehat{x}Q \rrbracket^{\lambda}_{\mathrm{N}} \;\triangleq\; (\lambda\alpha.\,\llbracket P \rrbracket^{\lambda}_{\mathrm{N}})(\lambda x.\,\llbracket Q \rrbracket_{\mathrm{A}})$$

This will be used in the cases dealing with propagation of left-cuts.

In order to show that typeability is preserved by the interpretation, we need first to define a CBN-translation of types:

Definition 6.2 Give a type constant Ω, we define a CBN-interpretation of types, that is split in two independent parts, inductively defined by

$$\begin{array}{rcl}
\langle \phi \rangle^{l}_{\mathrm{N}} & = & \phi^{oo} \\
\langle A\rightarrow B \rangle^{l}_{\mathrm{N}} & = & \langle A \rangle^{l}_{\mathrm{N}}\rightarrow\langle B \rangle^{l\;oo}_{\mathrm{N}} \\[1ex]
\langle \phi \rangle^{r}_{\mathrm{N}} & = & \phi^{o} \\
\langle A\rightarrow B \rangle^{r}_{\mathrm{N}} & = & (\langle A \rangle^{r\;o}_{\mathrm{N}})\rightarrow\langle B \rangle^{r\;oo}_{\mathrm{N}}
\end{array}$$

We define $\langle \Gamma, x{:}A \rangle^{l}_{\mathrm{N}} = \langle \Gamma \rangle^{l}_{\mathrm{N}}, x{:}\langle A \rangle^{l}_{\mathrm{N}}$, and $\langle \alpha{:}A, D \rangle^{r}_{\mathrm{N}} = \alpha{:}\langle A \rangle^{r}_{\mathrm{N}}, \langle \Delta \rangle^{r}_{\mathrm{N}}$.

The following result links the two interpretations.

Lemma 6.3 $\langle A \rangle^{l}_{\mathrm{N}} = \langle A \rangle^{r\,o}_{\mathrm{N}}$

Using these interpretations, we can show:

Theorem 6.4 *If $P : \Gamma \vdash_{\mathcal{X}} \Delta$, then $\langle \Gamma \rangle^{l}_{\mathrm{N}}, \langle \Delta \rangle^{r}_{\mathrm{N}} \vdash_{\lambda} \llbracket P \rrbracket^{\lambda}_{\mathrm{N}} : \Omega$.*

We can show that reduction in $\mathcal{X}$ is modeled by equality after interpretation:

Theorem 6.5 *If $P \rightarrow_{\mathrm{N}} Q$, then $\llbracket P \rrbracket^{\lambda}_{\mathrm{N}} =_{\mathrm{N}} \llbracket Q \rrbracket^{\lambda}_{\mathrm{N}}$.*

Proof: By induction on the length of the reduction path. We just check the rules, of which we show the interesting cases.

$(\mathit{exp\text{-}imp})$: $(\widehat{y}P\widehat{\beta}\cdot\alpha)\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\gamma}\,[x]\,\widehat{z}R) \rightarrow (Q\widehat{\gamma} \dagger \widehat{y}P)\widehat{\beta} \dagger \widehat{z}R$

$$\begin{array}{rcl}
\llbracket (\widehat{y}P\widehat{\beta}\cdot\alpha)\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\gamma}\,[x]\,\widehat{z}R) \rrbracket^{\lambda}_{\mathrm{N}} & \triangleq & (\lambda x.x\,\lambda u.(\lambda z.\llbracket R \rrbracket^{\lambda}_{\mathrm{N}})(u\,\lambda\gamma.\llbracket Q \rrbracket^{\lambda}_{\mathrm{N}}))(\lambda\alpha.\alpha\,\lambda y\beta.\llbracket P \rrbracket^{\lambda}_{\mathrm{N}}) \\
& =_{\mathrm{N}} & (\lambda\alpha.\alpha\,\lambda y\beta.\llbracket P \rrbracket^{\lambda}_{\mathrm{N}})(\lambda u.(\lambda z.\llbracket R \rrbracket^{\lambda}_{\mathrm{N}})(u\,\lambda\gamma.\llbracket Q \rrbracket^{\lambda}_{\mathrm{N}})) \\
(\alpha \notin \llbracket P \rrbracket^{\lambda}_{\mathrm{N}}) & =_{\mathrm{N}} & (\lambda u.(\lambda z.\llbracket R \rrbracket^{\lambda}_{\mathrm{N}})(u\,\lambda\gamma.\llbracket Q \rrbracket^{\lambda}_{\mathrm{N}}))(\lambda y\beta.\llbracket P \rrbracket^{\lambda}_{\mathrm{N}}) \\
& =_{\mathrm{N}} & (\lambda z.\llbracket R \rrbracket^{\lambda}_{\mathrm{N}})((\lambda y\beta.\llbracket P \rrbracket^{\lambda}_{\mathrm{N}})(\lambda\gamma.\llbracket Q \rrbracket^{\lambda}_{\mathrm{N}})) \\
& =_{\mathrm{N}} & (\lambda z.\llbracket R \rrbracket^{\lambda}_{\mathrm{N}})(\lambda\beta.\llbracket P \rrbracket^{\lambda}_{\mathrm{N}}\langle \lambda\gamma.\llbracket Q \rrbracket^{\lambda}_{\mathrm{N}}/y \rangle) \\
& =_{\mathrm{N}} & (\lambda z.\llbracket R \rrbracket^{\lambda}_{\mathrm{N}})(\lambda\beta.(\lambda y.\llbracket P \rrbracket^{\lambda}_{\mathrm{N}})(\lambda\gamma.\llbracket Q \rrbracket^{\lambda}_{\mathrm{N}})) \\
& \triangleq & \llbracket (Q\widehat{\gamma} \dagger \widehat{y}P)\widehat{\beta} \dagger \widehat{z}R \rrbracket^{\lambda}_{\mathrm{N}}
\end{array}$$

$$\begin{array}{rcl}
\text{Also:} & & (\lambda z.\llbracket R \rrbracket^{\lambda}_{\mathrm{N}})(\lambda\beta.\llbracket P \rrbracket^{\lambda}_{\mathrm{N}}\langle \lambda\gamma.\llbracket Q \rrbracket^{\lambda}_{\mathrm{N}}/y \rangle) \\
& =_{\mathrm{N}} & \llbracket R \rrbracket^{\lambda}_{\mathrm{N}}\langle \lambda\beta.\llbracket P \rrbracket^{\lambda}_{\mathrm{N}}\langle \lambda\gamma.\llbracket Q \rrbracket^{\lambda}_{\mathrm{N}}/y \rangle/z \rangle \\
(y \notin R, \beta \notin Q) & = & \llbracket R \rrbracket^{\lambda}_{\mathrm{N}}\langle \lambda\beta.\llbracket P \rrbracket^{\lambda}_{\mathrm{N}}/z \rangle\langle \lambda\gamma.\llbracket Q \rrbracket^{\lambda}_{\mathrm{N}}/y \rangle \\
& =_{\mathrm{N}} & (\lambda y.\llbracket R \rrbracket^{\lambda}_{\mathrm{N}}\langle \lambda\beta.\llbracket P \rrbracket^{\lambda}_{\mathrm{N}}/z \rangle)(\lambda\gamma.\llbracket Q \rrbracket^{\lambda}_{\mathrm{N}}) \\
& =_{\mathrm{N}} & (\lambda y.(\lambda z.\llbracket R \rrbracket^{\lambda}_{\mathrm{N}})(\lambda\beta.\llbracket P \rrbracket^{\lambda}_{\mathrm{N}}))(\lambda\gamma.\llbracket Q \rrbracket^{\lambda}_{\mathrm{N}}) \\
& \triangleq & \llbracket Q\widehat{\gamma} \dagger \widehat{y}(P\widehat{\beta} \dagger \widehat{z}R) \rrbracket^{\lambda}_{\mathrm{N}}
\end{array}$$

$(a\not\dagger)$: Notice that either $Q = \langle x\cdot\beta \rangle$, or $Q = Q_1\widehat{\beta}\ [x]\ \widehat{y}Q_2$, with $x \notin \mathit{fs}(Q_1,Q_2)$, so we can use $\llbracket Q \rrbracket^{\lambda}_{\mathrm{N}} \triangleq x\,\lambda u.\llbracket Q \rrbracket_{\mathrm{A}}[u/x]$, in all cases below.

$$\begin{array}{rcl}
\llbracket P\widehat{\alpha} \dagger \widehat{x}Q \rrbracket^{\lambda}_{\mathrm{N}} & \triangleq & (\lambda x.\llbracket Q \rrbracket^{\lambda}_{\mathrm{N}})(\lambda\alpha.\llbracket P \rrbracket^{\lambda}_{\mathrm{N}}) \\
& =_{\mathrm{N}} & \llbracket Q \rrbracket^{\lambda}_{\mathrm{N}}\langle \lambda\alpha.\llbracket P \rrbracket^{\lambda}_{\mathrm{N}}/x \rangle \\
& \triangleq & (x\,\lambda u.\llbracket Q \rrbracket_{\mathrm{A}}[u/x])\langle \lambda\alpha.\llbracket P \rrbracket^{\lambda}_{\mathrm{N}}/x \rangle \\
& = & (\lambda\alpha.\llbracket P \rrbracket^{\lambda}_{\mathrm{N}})(\lambda u.\llbracket Q \rrbracket_{\mathrm{A}}[u/x]) \\
(\alpha) & =_{\mathrm{N}} & (\lambda\alpha.\llbracket P \rrbracket^{\lambda}_{\mathrm{N}})(\lambda x.\llbracket Q \rrbracket_{\mathrm{A}}) \\
& \triangleq & (\lambda\alpha.\llbracket P \rrbracket^{\lambda}_{\mathrm{N}})(\lambda x.\llbracket Q \rrbracket_{\mathrm{A}}) \\
& \triangleq & \llbracket P\widehat{\alpha} \not\dagger \widehat{x}Q \rrbracket^{\lambda}_{\mathrm{N}}
\end{array}$$

$(d\not\dagger)$: $\langle y\cdot\alpha \rangle\widehat{\alpha} \not\dagger \widehat{x}P \rightarrow \langle y\cdot\alpha \rangle\widehat{\alpha} \dagger \widehat{x}P$.

$$\begin{array}{rcl}
\llbracket \langle y\cdot\alpha \rangle\widehat{\alpha} \not\dagger \widehat{x}P \rrbracket^{\lambda}_{\mathrm{N}} & \triangleq & (\lambda\alpha.y\lambda u.\alpha u)(\lambda x.\llbracket P \rrbracket_{\mathrm{A}}) \\
& =_{\mathrm{N}} & y\lambda u.(\lambda x.\llbracket P \rrbracket_{\mathrm{A}})u \\
& =_{\mathrm{N}} & (\lambda\alpha.y\lambda u.\alpha u)(\lambda x.\llbracket P \rrbracket_{\mathrm{A}}) \\
& =_{\mathrm{N}} & (\lambda v.v\lambda x.\llbracket P \rrbracket_{\mathrm{A}})(\lambda\alpha.y\lambda u.\alpha u) \\
(\alpha) & =_{\mathrm{N}} & (\lambda x.x\lambda v.\llbracket P \rrbracket_{\mathrm{A}}[v/x])(\lambda\alpha.y\lambda u.\alpha u) \\
& \triangleq & (\lambda x.\llbracket P \rrbracket^{\lambda}_{\mathrm{N}})(\lambda\alpha.y\lambda u.\alpha u) \\
& \triangleq & \llbracket \langle y\cdot\alpha \rangle\widehat{\alpha} \dagger \widehat{x}P \rrbracket^{\lambda}_{\mathrm{N}}
\end{array}$$

$(\mathit{exp\text{-}out}\not\dagger)$: $(\widehat{y}Q\widehat{\beta}\cdot\alpha)\widehat{\alpha} \not\dagger \widehat{x}P \rightarrow (\widehat{y}(Q\widehat{\alpha} \not\dagger \widehat{x}P)\widehat{\beta}\cdot\delta)\widehat{\delta} \dagger \widehat{x}P$, *$\gamma$ fresh*

$$\begin{array}{rrcl}
& \llbracket (\widehat{y}Q\widehat{\beta}\cdot\alpha)\widehat{\alpha} \nearrow \widehat{x}P \rrbracket_{\mathrm{N}}^{\lambda} & \triangleq & (\lambda\alpha.\alpha\,\lambda y\beta.\,\llbracket Q\rrbracket_{\mathrm{N}}^{\lambda})(\lambda x.\,\llbracket P\rrbracket_{\mathrm{A}}) \\
& & =_{\mathrm{N}} & (\alpha\,\lambda y\beta.\,\llbracket Q\rrbracket_{\mathrm{N}}^{\lambda})\langle \lambda x.\,\llbracket P\rrbracket_{\mathrm{A}}/\alpha\rangle \\
& & = & (\lambda x.\,\llbracket P\rrbracket_{\mathrm{A}})(\lambda y\beta.\,\llbracket Q\rrbracket_{\mathrm{N}}^{\lambda}\langle \lambda x.\,\llbracket P\rrbracket_{\mathrm{A}}/\alpha\rangle) \\
& & =_{\mathrm{N}} & (\lambda x.\,\llbracket P\rrbracket_{\mathrm{A}})(\lambda y\beta.(\lambda\alpha.\,\llbracket Q\rrbracket_{\mathrm{N}}^{\lambda})(\lambda x.\,\llbracket P\rrbracket_{\mathrm{A}})) \\
& & =_{\mathrm{N}} & (\lambda\delta.\delta\,\lambda y\beta.(\lambda\alpha.\,\llbracket Q\rrbracket_{\mathrm{N}}^{\lambda})(\lambda x.\,\llbracket P\rrbracket_{\mathrm{A}}))(\lambda x.\,\llbracket P\rrbracket_{\mathrm{A}}) \\
& & =_{\mathrm{N}} & (\lambda u.u\,\lambda x.\,\llbracket P\rrbracket_{\mathrm{A}})(\lambda\delta.\delta\,\lambda y\beta.(\lambda\alpha.\,\llbracket Q\rrbracket_{\mathrm{N}}^{\lambda})(\lambda x.\,\llbracket P\rrbracket_{\mathrm{A}})) \\
(\alpha) & & =_{\mathrm{N}} & (\lambda x.x\lambda u.\,\llbracket P\rrbracket_{\mathrm{A}}[u/x])(\lambda\delta.\delta\,\lambda y\beta.(\lambda\alpha.\,\llbracket Q\rrbracket_{\mathrm{N}}^{\lambda})(\lambda x.\,\llbracket P\rrbracket_{\mathrm{A}})) \\
& & =_{\mathrm{N}} & (\lambda x.\,\llbracket P\rrbracket_{\mathrm{N}}^{\lambda})(\lambda\delta.\delta\,\lambda y\beta.(\lambda\alpha.\,\llbracket Q\rrbracket_{\mathrm{N}}^{\lambda})(\lambda x.\,\llbracket P\rrbracket_{\mathrm{A}})) \\
& & \triangleq & \llbracket (\widehat{y}(Q\widehat{\alpha} \nearrow \widehat{x}P)\widehat{\beta}\cdot\delta)\widehat{\delta} \dagger \widehat{x}P \rrbracket_{\mathrm{N}}^{\lambda}
\end{array}$$

6.2 Call by Value

The results obtained above can be repeated for the CBV-subreduction. Again, note that, in the CBV reduction system, a right-cut $P\widehat{\alpha} \nwarrow \widehat{x}Q$ is only generated if P introduces α, so if $P = \langle y\cdot\alpha\rangle$, or $P = \widehat{y}R\widehat{\beta}\cdot\alpha$, and α not free in R.

The CBV-interpretation of nets in $\mathcal{X}$ as terms in λ is defined as follows:

Definition 6.6 (CBV Interpretation)

$$\begin{array}{rcl}
\llbracket \langle x\cdot\alpha\rangle \rrbracket_{\mathrm{V}}^{\lambda} & \triangleq & \alpha\,x \\
\llbracket \widehat{x}P\widehat{\alpha}\cdot\beta \rrbracket_{\mathrm{V}}^{\lambda} & \triangleq & \beta\,\lambda\alpha x.\,\llbracket P\rrbracket_{\mathrm{V}}^{\lambda} \\
\llbracket P\widehat{\alpha}\,[y]\,\widehat{x}Q \rrbracket_{\mathrm{V}}^{\lambda} & \triangleq & (\lambda\alpha.\,\llbracket P\rrbracket_{\mathrm{V}}^{\lambda})(y\,\lambda x.\,\llbracket Q\rrbracket_{\mathrm{V}}^{\lambda}) \\
\llbracket P\widehat{\alpha} \dagger \widehat{x}Q \rrbracket_{\mathrm{V}}^{\lambda} = \llbracket P\widehat{\alpha} \nearrow \widehat{x}Q \rrbracket_{\mathrm{V}}^{\lambda} & \triangleq & (\lambda\alpha.\,\llbracket P\rrbracket_{\mathrm{V}}^{\lambda})(\lambda x.\,\llbracket Q\rrbracket_{\mathrm{V}}^{\lambda}) \\
\llbracket \langle y\cdot\alpha\rangle\widehat{\alpha} \nwarrow \widehat{x}Q \rrbracket_{\mathrm{V}}^{\lambda} & \triangleq & (\lambda x.\,\llbracket Q\rrbracket_{\mathrm{V}}^{\lambda})\,y \\
\llbracket (\widehat{y}P\widehat{\beta}\cdot\alpha)\widehat{\alpha} \nwarrow \widehat{x}Q \rrbracket_{\mathrm{V}}^{\lambda} & \triangleq & (\lambda x.\,\llbracket Q\rrbracket_{\mathrm{V}}^{\lambda})(\lambda\beta y.\,\llbracket P\rrbracket_{\mathrm{V}}^{\lambda})
\end{array}$$

Again, defining

$$\begin{array}{rcl}
\llbracket \langle x\cdot\beta\rangle \rrbracket_{\mathrm{A}} & = & x \\
\llbracket \widehat{y}R\widehat{\beta}\cdot\alpha \rrbracket_{\mathrm{A}} & = & \lambda\beta y.\,\llbracket R\rrbracket_{\mathrm{V}}^{\lambda}
\end{array}$$

for the P that appears in $P\widehat{\alpha} \nwarrow \widehat{x}Q$, we have

$$\llbracket P\rrbracket_{\mathrm{V}}^{\lambda} \;=\; \alpha\,\llbracket P\rrbracket_{\mathrm{A}} \quad \textit{where } P \textit{ introduces } \alpha$$

Notice that $\llbracket P\rrbracket_{\mathrm{A}}$ is a value in those cases, which is important below to make sure that the reduction is call-by-value. We could have defined

$$\llbracket P\widehat{\alpha} \nwarrow \widehat{x}Q \rrbracket_{\mathrm{V}}^{\lambda} \;\triangleq\; (\lambda x.\,\llbracket Q\rrbracket_{\mathrm{V}}^{\lambda})\,\llbracket P\rrbracket_{\mathrm{A}}$$

This will be used in the proofs below for the cases dealing with propagation of right-cuts.

In order to show that typeability is preserved by the interpretation, we need first to define a CBV-translation of types:

Definition 6.7 Give again a type constant Ω, we now define a CBN-interpretation of types, that is also split in two independent parts, inductively by

$$\begin{array}{rcl}
\langle \varphi\rangle_{\mathrm{V}}^{l} & = & \varphi \\
\langle A{\rightarrow}B\rangle_{\mathrm{V}}^{l} & = & (\langle B\rangle_{\mathrm{V}}^{l\;o}){\rightarrow}\langle A\rangle_{\mathrm{V}}^{l\;o} \\[1ex]
\langle \varphi\rangle_{\mathrm{V}}^{r} & = & \varphi^{o} \\
\langle A{\rightarrow}B\rangle_{\mathrm{V}}^{r} & = & \langle B\rangle_{\mathrm{V}}^{r}{\rightarrow}\langle A\rangle_{\mathrm{V}}^{r\;o}
\end{array}$$

Again, we define $\langle \Gamma, x{:}A\rangle_{\mathrm{V}}^{l} = \langle \Gamma\rangle_{\mathrm{V}}^{l}, x{:}\langle A\rangle_{\mathrm{V}}^{l}$, and $\langle \alpha{:}A, D\rangle_{\mathrm{V}}^{r} = \alpha{:}\langle A\rangle_{\mathrm{V}}^{l}, \langle \Delta\rangle_{\mathrm{V}}^{l}$.

The following result links the two interpretations.

Lemma 6.8 $\langle A \rangle^{r}_{\mathrm{V}} = \langle A \rangle^{l\;o}_{\mathrm{V}}$.

Using these interpretations, we can show:

Theorem 6.9 *If $P : \Gamma \vdash_{\mathcal{X}} \Delta$, then $\langle \Gamma \rangle^{l}_{\mathrm{V}}, \langle \Delta \rangle^{r}_{\mathrm{V}} \vdash_{\lambda} \llbracket P \rrbracket^{\lambda}_{\mathrm{V}} : \Omega$.*

We can show that, also for the cbv-interpretation, reduction in $\mathcal{X}$ is modeled by equality after interpretation:

Theorem 6.10 *If $P \rightarrow_{\mathrm{v}} Q$, then $\llbracket P \rrbracket^{\lambda}_{\mathrm{V}} =_{\mathrm{v}} \llbracket Q \rrbracket^{\lambda}_{\mathrm{V}}$.*

Proof: By induction on the length of the reduction path. We just check the rules.
$(\textit{exp-imp})$:

$$
\begin{array}{rcl}
\llbracket (\widehat{y} P \widehat{\beta} \cdot \alpha) \widehat{\alpha} \dagger \widehat{x} (Q \widehat{\gamma} [x] \widehat{z} R) \rrbracket^{\lambda}_{\mathrm{V}} & \stackrel{\Delta}{=} & (\lambda \alpha . \alpha \, \lambda \beta y . \llbracket P \rrbracket^{\lambda}_{\mathrm{V}}) (\lambda x . (\lambda \gamma . \llbracket Q \rrbracket^{\lambda}_{\mathrm{V}}) (x \, \lambda z . \llbracket R \rrbracket^{\lambda}_{\mathrm{V}})) \\
(\alpha \notin \llbracket P \rrbracket^{\lambda}_{\mathrm{V}}) & =_{\mathrm{v}} & (\lambda x . (\lambda \gamma . \llbracket Q \rrbracket^{\lambda}_{\mathrm{V}}) (x \, \lambda z . \llbracket R \rrbracket^{\lambda}_{\mathrm{V}})) (\lambda \beta . \lambda y . \llbracket P \rrbracket^{\lambda}_{\mathrm{V}}) \\
 & =_{\mathrm{v}} & (\lambda \gamma . \llbracket Q \rrbracket^{\lambda}_{\mathrm{V}}) ((\lambda \beta y . \llbracket P \rrbracket^{\lambda}_{\mathrm{V}}) (\lambda z . \llbracket R \rrbracket^{\lambda}_{\mathrm{V}})) \\
 & =_{\mathrm{v}} & (\lambda \gamma . \llbracket Q \rrbracket^{\lambda}_{\mathrm{V}}) (\lambda y . (\, \llbracket P \rrbracket^{\lambda}_{\mathrm{V}} \langle \lambda z . \llbracket R \rrbracket^{\lambda}_{\mathrm{V}} / \beta \rangle)) \\
 & =_{\mathrm{v}} & (\lambda \gamma . \llbracket Q \rrbracket^{\lambda}_{\mathrm{V}}) (\lambda y . (\lambda \beta . \llbracket P \rrbracket^{\lambda}_{\mathrm{V}}) (\lambda z . \llbracket R \rrbracket^{\lambda}_{\mathrm{V}})) \\
 & \stackrel{\Delta}{=} & \llbracket Q \widehat{\gamma} \dagger \widehat{y} (P \widehat{\beta} \dagger \widehat{z} R) \rrbracket^{\lambda}_{\mathrm{V}}
\end{array}
$$

$$
\begin{array}{rcl}
\text{Also:} & & (\lambda \gamma . \llbracket Q \rrbracket^{\lambda}_{\mathrm{V}}) (\lambda y . \llbracket P \rrbracket^{\lambda}_{\mathrm{V}} \langle \lambda z . \llbracket R \rrbracket^{\lambda}_{\mathrm{V}} / \beta \rangle) \\
 & =_{\mathrm{v}} & \llbracket Q \rrbracket^{\lambda}_{\mathrm{V}} \langle \lambda y . \llbracket P \rrbracket^{\lambda}_{\mathrm{V}} \langle \lambda z . \llbracket R \rrbracket^{\lambda}_{\mathrm{V}} / \beta \rangle / \gamma \rangle \\
(y \notin R, \beta \notin Q) & = & \llbracket Q \rrbracket^{\lambda}_{\mathrm{V}} \langle \lambda y . \llbracket P \rrbracket^{\lambda}_{\mathrm{V}} / \gamma \rangle \langle \lambda z . \llbracket R \rrbracket^{\lambda}_{\mathrm{V}} / \beta \rangle \\
 & =_{\mathrm{v}} & (\lambda \beta . \llbracket Q \rrbracket^{\lambda}_{\mathrm{V}} \langle \lambda y . \llbracket P \rrbracket^{\lambda}_{\mathrm{V}} / \gamma \rangle) (\lambda z . \llbracket R \rrbracket^{\lambda}_{\mathrm{V}}) \\
 & =_{\mathrm{v}} & (\lambda \beta . (\lambda \gamma . \llbracket Q \rrbracket^{\lambda}_{\mathrm{V}}) (\lambda y . \llbracket P \rrbracket^{\lambda}_{\mathrm{V}})) (\lambda z . \llbracket R \rrbracket^{\lambda}_{\mathrm{V}}) \\
 & \stackrel{\Delta}{=} & (\lambda \beta . \llbracket Q \widehat{\gamma} \dagger \widehat{y} P \rrbracket^{\lambda}_{\mathrm{V}}) (\lambda z . \llbracket R \rrbracket^{\lambda}_{\mathrm{V}}) \\
 & \stackrel{\Delta}{=} & \llbracket (Q \widehat{\gamma} \dagger \widehat{y} P) \widehat{\beta} \dagger \widehat{z} R \rrbracket^{\lambda}_{\mathrm{V}}
\end{array}
$$

$(\nwarrow a)$: $P \widehat{\alpha} \dagger \widehat{x} Q \rightarrow P \widehat{\alpha} \nwarrow \widehat{x} Q$, x not introduced.
Notice that either $P = \langle y \cdot \alpha \rangle$, or $P = \widehat{y} R \widehat{\beta} \cdot \alpha$, with $\alpha \notin fp(R)$, so we can use $\llbracket P \rrbracket^{\lambda}_{\mathrm{V}} \stackrel{\Delta}{=} \alpha \, \llbracket P \rrbracket_{\mathrm{A}}$, in all cases below.

$$
\begin{array}{rcl}
\llbracket \langle y \cdot \alpha \rangle \widehat{\alpha} \dagger \widehat{x} Q \rrbracket^{\lambda}_{\mathrm{V}} & \stackrel{\Delta}{=} & (\lambda \alpha . \llbracket \langle y \cdot \alpha \rangle \rrbracket^{\lambda}_{\mathrm{V}}) (\lambda x . \llbracket Q \rrbracket^{\lambda}_{\mathrm{V}}) \\
 & \stackrel{\Delta}{=} & (\lambda \alpha . \llbracket P \rrbracket^{\lambda}_{\mathrm{V}}) (\lambda x . \llbracket Q \rrbracket^{\lambda}_{\mathrm{V}}) \\
 & \stackrel{\Delta}{=} & (\lambda \alpha . \alpha \, \llbracket P \rrbracket_{\mathrm{A}}) (\lambda x . \llbracket Q \rrbracket^{\lambda}_{\mathrm{V}}) \\
 & =_{\mathrm{v}} & (\lambda x . \llbracket Q \rrbracket^{\lambda}_{\mathrm{V}}) \; \llbracket P \rrbracket_{\mathrm{A}} \\
 & \stackrel{\Delta}{=} & \llbracket P \widehat{\alpha} \nwarrow \widehat{x} Q \rrbracket^{\lambda}_{\mathrm{V}}
\end{array}
$$

$(\nwarrow d)$:

$$
\begin{array}{rcl}
\llbracket P \widehat{\alpha} \nwarrow \widehat{x} \langle x \cdot \beta \rangle \rrbracket^{\lambda}_{\mathrm{V}} & \stackrel{\Delta}{=} & (\lambda x . \llbracket \langle x \cdot \beta \rangle \rrbracket^{\lambda}_{\mathrm{V}}) \; \llbracket P \rrbracket_{\mathrm{A}} \\
 & =_{\mathrm{v}} & (\lambda \alpha . \alpha \; \llbracket P \rrbracket_{\mathrm{A}}) (\lambda x . \llbracket \langle x \cdot \beta \rangle \rrbracket^{\lambda}_{\mathrm{V}}) \\
 & = & (\lambda \alpha . \llbracket P \rrbracket^{\lambda}_{\mathrm{V}}) (\lambda x . \llbracket \langle x \cdot \beta \rangle \rrbracket^{\lambda}_{\mathrm{V}}) \\
 & \stackrel{\Delta}{=} & \llbracket P \widehat{\alpha} \dagger \widehat{x} \langle x \cdot \beta \rangle \rrbracket^{\lambda}_{\mathrm{V}}
\end{array}
$$

$(\nwarrow \textit{imp-out})$:

$$
\begin{array}{rcl}
\llbracket P\widehat{\alpha} \nwarrow \widehat{x}(Q\widehat{v}\,[x]\,\widehat{y}R)\rrbracket^{\lambda}_{\mathrm{V}} & \triangleq & (\lambda x.\llbracket Q\widehat{v}\,[x]\,\widehat{y}R\rrbracket^{\lambda}_{\mathrm{V}})\,\llbracket P\rrbracket_{\mathrm{A}} \\
& \triangleq & (\lambda x.(\lambda v.\llbracket Q\rrbracket^{\lambda}_{\mathrm{V}})(x\,\lambda y.\llbracket R\rrbracket^{\lambda}_{\mathrm{V}}))\,\llbracket P\rrbracket_{\mathrm{A}} \\
& =_{\mathrm{V}} & (\lambda v.(\llbracket Q\rrbracket^{\lambda}_{\mathrm{V}}\langle \llbracket P\rrbracket_{\mathrm{A}}/x\rangle))(\llbracket P\rrbracket_{\mathrm{A}}\,\lambda y.(\llbracket R\rrbracket^{\lambda}_{\mathrm{V}}\langle \llbracket P\rrbracket_{\mathrm{A}}/x\rangle)) \\
& =_{\mathrm{V}} & (\lambda v.((\lambda x.\llbracket Q\rrbracket^{\lambda}_{\mathrm{V}})\,\llbracket P\rrbracket_{\mathrm{A}}))(\llbracket P\rrbracket_{\mathrm{A}}\,\lambda y.((\lambda x.\llbracket R\rrbracket^{\lambda}_{\mathrm{V}})\,\llbracket P\rrbracket_{\mathrm{A}})) \\
& =_{\mathrm{V}} & (\lambda z.(\lambda v.((\lambda x.\llbracket Q\rrbracket^{\lambda}_{\mathrm{V}})\,\llbracket P\rrbracket_{\mathrm{A}}))(z\,\lambda y.((\lambda x.\llbracket R\rrbracket^{\lambda}_{\mathrm{V}})\,\llbracket P\rrbracket_{\mathrm{A}})))\,\llbracket P\rrbracket_{\mathrm{A}} \\
& =_{\mathrm{V}} & (\lambda\alpha.\alpha\,\llbracket P\rrbracket_{\mathrm{A}})(\lambda z.(\lambda v.\llbracket P\widehat{\alpha} \nwarrow \widehat{x}Q\rrbracket^{\lambda}_{\mathrm{V}})(z\,\lambda y.\llbracket P\widehat{\alpha} \nwarrow \widehat{x}R\rrbracket^{\lambda}_{\mathrm{V}})) \\
& = & (\lambda\alpha.\llbracket P\rrbracket^{\lambda}_{\mathrm{V}})(\lambda z.(\lambda v.\llbracket P\widehat{\alpha} \nwarrow \widehat{x}Q\rrbracket^{\lambda}_{\mathrm{V}})(z\,\lambda y.\llbracket P\widehat{\alpha} \nwarrow \widehat{x}R\rrbracket^{\lambda}_{\mathrm{V}})) \\
& \triangleq & (\lambda\alpha.\llbracket P\rrbracket^{\lambda}_{\mathrm{V}})(\lambda z.\llbracket (P\widehat{\alpha} \nwarrow \widehat{x}Q)\widehat{v}\ [z]\ \widehat{y}(P\widehat{\alpha} \nwarrow \widehat{x}R)\rrbracket^{\lambda}_{\mathrm{V}}) \\
& \triangleq & \llbracket P\widehat{\alpha} \dagger \widehat{z}((P\widehat{\alpha} \nwarrow \widehat{x}Q)\widehat{v}\ [z]\ \widehat{y}(P\widehat{\alpha} \nwarrow \widehat{x}R))\rrbracket^{\lambda}_{\mathrm{V}}
\end{array}
$$

$(\textit{exp-out} \nearrow)$:

$$
\begin{array}{rcl}
\llbracket (\widehat{y}Q\widehat{\beta}\cdot\alpha)\widehat{\alpha} \nearrow \widehat{x}P\rrbracket^{\lambda}_{\mathrm{V}} & \triangleq & (\lambda\alpha.\alpha\,\lambda\beta y.\llbracket Q\rrbracket^{\lambda}_{\mathrm{V}})(\lambda x.\llbracket P\rrbracket^{\lambda}_{\mathrm{V}}) \\
& =_{\mathrm{V}} & (\alpha\,\lambda\beta y.\llbracket Q\rrbracket^{\lambda}_{\mathrm{V}})\langle \lambda x.\llbracket P\rrbracket^{\lambda}_{\mathrm{V}}/\alpha\rangle \\
& =_{\mathrm{V}} & (\lambda x.\llbracket P\rrbracket^{\lambda}_{\mathrm{V}})\,\lambda\beta y.(\llbracket Q\rrbracket^{\lambda}_{\mathrm{V}}\langle \lambda x.\llbracket P\rrbracket^{\lambda}_{\mathrm{V}}/\alpha\rangle) \\
& =_{\mathrm{V}} & (\lambda\delta.\delta\,\lambda\beta y.(\lambda\alpha.\llbracket Q\rrbracket^{\lambda}_{\mathrm{V}})(\lambda x.\llbracket P\rrbracket^{\lambda}_{\mathrm{V}}))(\lambda x.\llbracket P\rrbracket^{\lambda}_{\mathrm{V}}) \\
& \triangleq & \llbracket (\widehat{y}(Q\widehat{\alpha} \nearrow \widehat{x}P)\widehat{\beta}\cdot\delta)\widehat{\delta} \dagger \widehat{x}P\rrbracket^{\lambda}_{\mathrm{V}}
\end{array}
$$

7 From $\mathcal{X}$ to $\lambda\mu$

We have seen above, for the pure λ-calculus, that besides the fact that we translate a symmetric calculus to a strongly right-oriented one, reflecting the orientation of propagation rules requires some particular attention in the translation. In this section, we will show that the results achieved above are obtainable for $\lambda\mu$ as well, first for Call by Name, and then for Call by Value. However, the details of the interpretations will differ significantly.

The $\lambda\mu$-calculus is expressive enough to allow for the preservation of types. Its asymmetric nature will prove to be no obstacle; it is overcome by splitting the interpretation into two steps. We perform first a compositional translation into $\lambda\mu$ which mimics the double negation translation in the sense of Plotkin's CBN, and thus alter the types as well. The second step consists in recovering the types, which can only be done globally. This first result implies that we can faithfully interpret $\mathcal{X}$ into $\lambda\mu$, the second that typeability is fully preserved.

A natural idea would be to start from our former λ-calculus interpretations (see previous section) to build this interpretation. However, this is not going to help much, as there the two kinds of variables have been merged. Instead, we need to handle circuits directly, which means we are actually working out a genuine new interpretation.

Let us focus on the CBN-interpretation; the CBV one comes along very similar guidelines. Towards our main theorem, sockets are translated into ordinary λ-variables, and plugs are embedded into μ-variables; for both, we keep the same name, to ease the reading. We interpret both ordinary and right cuts the same way, while the left cut still requires switching the position of the sub-terms, as the reduction rules in the (standard) $\lambda\mu$-calculus are intrinsically call-by-name. Therefore, the flipping mechanism, used above to left-activate a cut or to de-activate a left-cut, is still required for the interpretation to hold along the bi-directional propagation rules from $\mathcal{X}$.

The formal definitions given below should be self-explanatory, except from the observation that a double-negation step is still used. This point deserves an explanation. The circuit $P\widehat{\alpha} \dagger \widehat{x}Q$ for which we know

$$P\widehat{\alpha} \dagger \widehat{x}Q : \Gamma \vdash_{\mathcal{X}} \Delta$$

is translated into a command $[\omega](\lambda x.M)\,(\mu\alpha.C)$ for some $\lambda\mu$-term M and command C and some μ-variable ω. The translation cannot be an ordinary term for both a syntactic reason and a semantic one. Since we target the standard $\lambda\mu$-calculus, the interpretation of the sub-circuit P has to be a command, otherwise the μ-binding cannot be defined; as a matter of consequence, the translation of Q is going to be of the shape $[\omega']M'$, which means that $M = \mu\eta.[\omega']M'$ for some μ-variables η, ω'.

From the $\lambda\mu$-calculus perspective, this observation means we are switching contexts intensively. This provides actually a semantic view on the way the $\mathcal{X}$-calculus manages derivations trees. The cut shows no privileged assumption nor conclusion, just connecting P and Q circuits. The lack of a selected assumption is reflected by the interpretation not to be an abstraction; also, the lack of any particular conclusion has the consequence we cannot use a μ binder at this point. Therefore, cuts in $\mathcal{X}$ correspond to proofs handled at the level of commands in $\lambda\mu$.

At this stage, we could have choose to treat commands as proofs for the $\perp$ negation type [11, 8]. This would have been a fair choice, since $\lambda\mu$ allows us to build a proof in any type from such a proof. However, the translation itself can carry out a more elegant and precise information. The definition that we propose shows that, given an arbitrary, initial choice of a named conclusion $\omega : \Omega$, building circuits into $\mathcal{X}$ can be read as syntax rules for dealing with judgments, by switching over the targeted conclusion Ω as a *pivot*. (So we have $\omega = \omega'$ in the previous informal analysis, and the formal definition.)

As a matter of consequence, we get that the interpretation of any given proof net can be built by targeting one conclusion $\alpha{:}A$ among its non-empty set of conclusions, leading to a $\lambda\mu$-term M which expresses the view that the proof for A is done by switching back and forth with the conclusion.

Paradigmatic examples of such proofs are provided by Peirce's law or the *ex falso quodlibet* formula $\neg\neg T \to T$. Our interpretation shows that this mechanism is general, at least as far as the restricted fragment of sequent calculus $\mathcal{X}$ deals with can tell.

7.1 Call by Name

As for the case of the λ-calculus, our results will depend not only on an interpretation of terms, but also on one for types. Let Ω be, as before, any (fixed) type and denote $\neg T \equiv T{\to}\Omega$ for convenience. The following lemma will provide the necessary trick.

Take

$$\begin{array}{rcccl} \mathit{force}\,F & \triangleq & \mu\tau.[\omega]F\,\lambda t.\mu!.[\tau]t & =_{\mathrm{v}} & F^{-} \\ \mathit{delay}\,t & \triangleq & \lambda f.f\,t & \triangleq & (t)^{\star} \end{array}$$

Lemma 7.1 For every type T, there exists

$$\begin{array}{rl} \mathit{force}: & \neg\neg T{\to}T \\ \mathit{delay}: & T{\to}\neg\neg T \end{array}$$

such that force $\circ$ delay is the identity on T (up to equality).

Proof: Given $t{:}T$, we get $\mathit{force}\,(t)^{\star} = \mu\tau.[\omega](t)^{\star}\,\lambda x.\mu!.[\tau]x = \mu\tau.[\tau]t = t$. In short $((t)^{\star})^{-} = t$. $\square$

Definition 7.2 (Plotkin's CBN) The CBN interpretation of type T, denoted $\ulcorner T\lrcorner^{\mu}_{\mathrm{N}} \triangleq \ulcorner T\lrcorner^{\mu}_{\mathrm{N}} \triangleq \neg\neg\ulcorner T\lrcorner^{\mu'}_{\mathrm{N}}$ is defined inductively by

$$\begin{array}{rcll} \ulcorner X\lrcorner^{\mu'}_{\mathrm{N}} & \triangleq & X, & X \textit{ type variable} \\ \ulcorner A{\to}B\lrcorner^{\mu'}_{\mathrm{N}} & \triangleq & \ulcorner A\lrcorner^{\mu}_{\mathrm{N}}{\to}\ulcorner B\lrcorner^{\mu}_{\mathrm{N}} & \end{array}$$

Also $\ulcorner \Gamma, x{:}T \lrcorner^{\mu}_{\mathbf{N}} \triangleq \ulcorner \Gamma \lrcorner^{\mu}_{\mathbf{N}}, x{:}\ulcorner T \lrcorner^{\mu}_{\mathbf{N}}$.

Type recovery is possible, owing to the following result.

Lemma 7.3 For any type T, there exist $\varphi_T : \ulcorner T \lrcorner^{\mu}_{\mathbf{N}} \to T$ and $\psi_T : T \to \ulcorner T \lrcorner^{\mu}_{\mathbf{N}}$.

Proof: Simple induction on T:

$(T \triangleq X)$: Take

$$\begin{array}{rcl} \varphi_X F & \triangleq & \mu\tau.[\omega] F\, \lambda t.\mu !.[\tau] t \\ \psi_X t & \triangleq & \lambda F.F\, t \end{array}$$

$(T \triangleq A \to B)$: Define

$$\begin{array}{rcl} \varphi_{A\to B} F & \triangleq & \lambda a.F\, \lambda f.\varphi_B\, f\, \psi_A\, a \\ \psi_{A\to B}\, t & \triangleq & \lambda F.F\, \lambda a.\psi_B\, t\, \varphi_A\, a \end{array}$$

$\square$

The first step of the interpretation is type-free, athough our definition aims at complying with types later.

The notation $\mu !.C$ is a shortcut for $\mu\eta.C$ where η is a fresh μ-variable wrt C.

Definition 7.4 (Call by name) Let

$$\begin{array}{rcl} \ulcorner \langle x{\cdot}\alpha \rangle \lrcorner^{\mu'}_{\mathbf{N}} & \triangleq & \lambda v.\mu !.[\alpha](v)^{\star} \\ \ulcorner P \widehat{\beta}\, [x]\, \widehat{y} Q \lrcorner^{\mu'}_{\mathbf{N}} & \triangleq & \lambda v.\mu !.[\omega]\lambda y.\mu !.\ulcorner Q \lrcorner^{\mu}_{\mathbf{N}}\, v\, \mu\beta.\ulcorner P \lrcorner^{\mu}_{\mathbf{N}} \end{array}$$

For P any $\mathcal{X}$-term, we define $\ulcorner P \lrcorner^{\mu}_{\mathbf{N}}$ by structural induction

$$\begin{array}{rcl} \ulcorner \langle x{\cdot}\alpha \rangle \lrcorner^{\mu}_{\mathbf{N}} & \triangleq & [\omega]\, x\, \ulcorner \langle x{\cdot}\alpha \rangle \lrcorner^{\mu'}_{\mathbf{N}} \\ \ulcorner \widehat{y} P \widehat{\beta}{\cdot}\alpha \lrcorner^{\mu}_{\mathbf{N}} & \triangleq & [\alpha](\lambda y.\mu\beta.\ulcorner P \lrcorner^{\mu}_{\mathbf{N}})^{\star} \\ \ulcorner P \widehat{\beta}\, [x]\, \widehat{y} Q \lrcorner^{\mu}_{\mathbf{N}} & \triangleq & [\omega]\, x\, \ulcorner P \widehat{\beta}\, [x]\, \widehat{y} Q \lrcorner^{\mu'}_{\mathbf{N}} \\ \ulcorner P \widehat{\alpha} \dagger \widehat{x} Q \lrcorner^{\mu}_{\mathbf{N}} \triangleq \ulcorner P \widehat{\alpha} \nwarrow \widehat{x} Q \lrcorner^{\mu}_{\mathbf{N}} & \triangleq & [\omega]\lambda x.\mu !.\ulcorner Q \lrcorner^{\mu}_{\mathbf{N}}\, \mu\alpha.\ulcorner P \lrcorner^{\mu}_{\mathbf{N}} \\ \ulcorner P \widehat{\alpha} \nearrow \widehat{x} Q \lrcorner^{\mu}_{\mathbf{N}} & \triangleq & [\omega]\mu\alpha.\ulcorner P \lrcorner^{\mu}_{\mathbf{N}}\, \ulcorner Q \lrcorner^{\mu'}_{\mathbf{N}} \end{array}$$

Proposition 7.5 (Conservation of types in cbn) *If $P : \Gamma \vdash_{\mathcal{X}} \Delta$, then $\ulcorner \Gamma \lrcorner^{\mu}_{\mathbf{N}} \vdash_{\lambda\mu} \ulcorner P \lrcorner^{\mu}_{\mathbf{N}} : cmd \mid \omega{:}\Omega, \ulcorner \Delta \lrcorner^{\mu}_{\mathbf{N}}$.*

Proof: By induction on the structure of nets.

$(\langle x{\cdot}\alpha \rangle)$: Then there exists T type such that $x{:}T$ and $\alpha{:}T$. Thus $\lambda v.\mu !.[\alpha](v)^{\star} : \neg \ulcorner A \lrcorner^{\mu'}_{\mathbf{N}}$ and $x\, \lambda v.\mu !.[\alpha](v)^{\star}{:}\Omega$as required.

$(\widehat{y} P \widehat{\beta}{\cdot}\alpha)$: By induction

$$\ulcorner \Gamma \lrcorner^{\mu}_{\mathbf{N}}, y{:}\ulcorner A \lrcorner^{\mu}_{\mathbf{N}} \vdash_{\lambda\mu} \ulcorner P \lrcorner^{\mu}_{\mathbf{N}} : cmd \mid \beta{:}\ulcorner B \lrcorner^{\mu}_{\mathbf{N}}, \alpha{:}\ulcorner A \to B \lrcorner^{\mu}_{\mathbf{N}}, \omega{:}\Omega, \ulcorner \Delta \lrcorner^{\mu}_{\mathbf{N}}$$

Then also

$$\ulcorner \Gamma \lrcorner^{\mu}_{\mathbf{N}} \vdash_{\lambda\mu} \lambda y.\mu\beta.\ulcorner P \lrcorner^{\mu}_{\mathbf{N}} : \ulcorner A \lrcorner^{\mu}_{\mathbf{N}} \to \ulcorner B \lrcorner^{\mu}_{\mathbf{N}} \mid \alpha{:}\ulcorner A \to B \lrcorner^{\mu}_{\mathbf{N}}, \omega{:}\Omega, \ulcorner \Delta \lrcorner^{\mu}_{\mathbf{N}}$$

Since $(M)^{\star} =_{\mathrm{v}} \lambda f.f\, M$ for any term M, we get

$$\ulcorner \Gamma \lrcorner^{\mu}_{\mathbf{N}} \vdash_{\lambda\mu} (\lambda y.\mu\beta.\ulcorner P \lrcorner^{\mu}_{\mathbf{N}})^{\star} : \ulcorner A \to B \lrcorner^{\mu}_{\mathbf{N}} \mid \alpha{:}\ulcorner A \to B \lrcorner^{\mu}_{\mathbf{N}}, \omega{:}\Omega, \ulcorner \Delta \lrcorner^{\mu}_{\mathbf{N}}$$

as expected.

$(P \widehat{\beta}\, [x]\, \widehat{y} Q)$: By induction

$$\ulcorner \Gamma \lrcorner^{\mu}_{\mathbf{N}}, x{:}\ulcorner A \to B \lrcorner^{\mu}_{\mathbf{N}} \vdash_{\lambda\mu} \mu\beta.\ulcorner P \lrcorner^{\mu}_{\mathbf{N}} : \ulcorner A \lrcorner^{\mu}_{\mathbf{N}} \mid \omega{:}\Omega, \ulcorner \Delta \lrcorner^{\mu}_{\mathbf{N}}$$

and

$$\ulcorner \Gamma \lrcorner^{\mu}_{\mathbf{N}}, x{:}\ulcorner A \to B \lrcorner^{\mu}_{\mathbf{N}} \vdash_{\lambda\mu} \lambda y.\mu !.\ulcorner Q \lrcorner^{\mu}_{\mathbf{N}} : \ulcorner B \lrcorner^{\mu}_{\mathbf{N}} \to \Omega \mid \omega{:}\Omega, \ulcorner \Delta \lrcorner^{\mu}_{\mathbf{N}}$$

Taking $v{:}\ulcorner A \lrcorner^{\mu}_{\mathrm{N}} \to \ulcorner B \lrcorner^{\mu}_{\mathrm{N}}$, we get

$$\ulcorner \Gamma \lrcorner^{\mu}_{\mathrm{N}}, x{:}\ulcorner A \to B \lrcorner^{\mu}_{\mathrm{N}} \vdash_{\lambda\mu} \ulcorner P \widehat{\beta} [x] \widehat{y} Q \lrcorner^{\mu}_{\mathrm{N}} : cmd \mid \omega{:}\Omega, \ulcorner \Delta \lrcorner^{\mu}_{\mathrm{N}}$$

$(P\widehat{\alpha} \dagger \widehat{x}Q, P\widehat{\alpha} \nwarrow \widehat{x}Q)$: By induction

$$\ulcorner \Gamma \lrcorner^{\mu}_{\mathrm{N}} \vdash_{\lambda\mu} \lambda x.\mu !.\ulcorner Q \lrcorner^{\mu}_{\mathrm{N}} : \ulcorner T \lrcorner^{\mu}_{\mathrm{N}} \to \Omega \mid \omega{:}\Omega, \ulcorner \Delta \lrcorner^{\mu}_{\mathrm{N}}$$

and

$$\ulcorner \Gamma \lrcorner^{\mu}_{\mathrm{N}} \vdash_{\lambda\mu} \mu\alpha.\ulcorner P \lrcorner^{\mu}_{\mathrm{N}} : \ulcorner T \lrcorner^{\mu}_{\mathrm{N}} \mid \omega{:}\Omega, \ulcorner \Delta \lrcorner^{\mu}_{\mathrm{N}}$$

Then

$$\ulcorner \Gamma \lrcorner^{\mu}_{\mathrm{N}} \vdash_{\lambda\mu} \ulcorner P\widehat{\alpha} \dagger \widehat{x}Q \lrcorner^{\mu}_{\mathrm{N}} : cmd \mid \omega{:}\Omega, \ulcorner \Delta \lrcorner^{\mu}_{\mathrm{N}}$$

$(P\widehat{\alpha} \not\nearrow \widehat{x}Q)$: The induction hypotheses are the same as for the previous part. We know x is introduced in Q, but whatever x, we get

$$\Gamma \vdash_{\lambda\mu} \lambda v.\lambda x.\mu !.\ulcorner Q \lrcorner^{\mu}_{\mathrm{N}} (v)^{\star} : \neg \ulcorner T \lrcorner^{\mu'}_{\mathrm{N}}$$

Since $\ulcorner T \lrcorner^{\mu}_{\mathrm{N}} = \neg\neg \ulcorner T \lrcorner^{\mu'}_{\mathrm{N}}$, we are done. □

Theorem 7.6 *For any $P : \Gamma \vdash_{\mathcal{X}} \Delta$ in $\mathcal{X}$, and type Ω there exists a $\lambda\mu$-term $\llbracket P \rrbracket^{\Omega}_{\mathrm{N}}$ such that $\Gamma \vdash_{\lambda\mu} \llbracket P \rrbracket^{\Omega}_{\mathrm{N}} : \Omega \mid \Delta$.*

Proof: Let ω a fresh μ-variable. The previous proposition provides $\ulcorner \Gamma \lrcorner^{\mu}_{\mathrm{N}} \vdash_{\lambda\mu} \ulcorner P \lrcorner^{\mu}_{\mathrm{N}} : cmd \mid \omega{:}\Omega, \ulcorner \Delta \lrcorner^{\mu}_{\mathrm{N}}$, hence $\ulcorner \Gamma \lrcorner^{\mu}_{\mathrm{N}} \vdash_{\lambda\mu} \mu !.[\omega] \ulcorner P \lrcorner^{\mu}_{\mathrm{N}} : \Omega \mid \ulcorner \Delta \lrcorner^{\mu}_{\mathrm{N}}$. We still need to take care of free variables. Assume the notation $\tilde{v}{:}\ulcorner T \lrcorner^{\mu}_{\mathrm{N}}$ for each $v{:}T$ in whatever context.

Since $\psi_T{:}T \to \ulcorner T \lrcorner^{\mu}_{\mathrm{N}}$, the case of λ-variables is easy: substitute each occurrence of $\tilde{x}$ by $\psi_T\, x$, which is simply the thunk $(x)^{\star}$ when T is atomic.

For $\alpha{:}T$, $\ulcorner P \lrcorner^{\mu}_{\mathrm{N}}$ may contains occurrences of commands such as $[\tilde{\alpha}]M$. We expect the command $[\alpha]\varphi_T\, M$ in turn: this is not possible through the substitution mechanism provided by μ-reduction. The term $\ulcorner P \lrcorner^{\mu'}_{\mathrm{N}\,\Omega}$ is thus obtained by make the replacements have been the meta-level. □

Our goal is that of Theorem 7.8. for which we first show a lemma.

Lemma 7.7 If x is introduced in P, then

$$\lambda u.\lambda x.\mu !.\ulcorner P \lrcorner^{\mu}_{\mathrm{N}} (u)^{\star} \quad =_{\mathrm{N}} \quad \ulcorner P \lrcorner^{\mu'}_{\mathrm{N}}$$

Proof: By hypothesis, P is either a capsule or an import, in which case x does not appear free in $\ulcorner P \lrcorner^{\mu'}_{\mathrm{N}}$. Therefore,

$$\begin{array}{lll}
(P \equiv \langle x{\cdot}\alpha\rangle): & \lambda u.\lambda x.\mu !.\ulcorner P \lrcorner^{\mu}_{\mathrm{N}} (u)^{\star} & =_{\mathrm{N}} \ \lambda u.\lambda x.\mu !.[\omega] x \ulcorner P \lrcorner^{\mu'}_{\mathrm{N}} (u)^{\star} \\
 & & =_{\mathrm{N}} \ \lambda u.\mu !.[\omega](u)^{\star} \ulcorner P \lrcorner^{\mu'}_{\mathrm{N}} \\
 & & =_{\mathrm{N}} \ \lambda u.\mu !.[\omega](u)^{\star} \lambda v.\mu !.[\alpha](v)^{\star} \\
 & & =_{\mathrm{N}} \ \lambda u.\mu !.[\omega] \lambda v.\mu !.[\alpha](v)^{\star}\, u \\
 & & =_{\mathrm{N}} \ \lambda u.\mu !.[\alpha](u)^{\star} \\
 & & =_{\mathrm{V}} \ \ulcorner P \lrcorner^{\mu'}_{\mathrm{N}}
\end{array}$$

$$\begin{array}{ll}
(P \equiv Q\widehat{\beta}[x]\widehat{z}R): & \lambda u.\lambda x.\mu !.\ulcorner P \lrcorner^{\mu}_{\mathrm{N}} (u)^{\star} \\
=_{\mathrm{N}} & \lambda u.\lambda x.\mu !.[\omega] x \ulcorner P \lrcorner^{\mu'}_{\mathrm{N}} (u)^{\star} \\
=_{\mathrm{N}} & \lambda u.\mu !.[\omega](u)^{\star} \ulcorner P \lrcorner^{\mu'}_{\mathrm{N}} \\
=_{\mathrm{N}} & \lambda u.\mu !.[\omega](u)^{\star} \lambda v.\mu !.[\omega] \lambda z.\mu !.\ulcorner R \lrcorner^{\mu}_{\mathrm{N}}\, v\, \mu\beta.\ulcorner Q \lrcorner^{\mu}_{\mathrm{N}} \\
=_{\mathrm{N}} & \lambda u.\mu !.[\omega] \lambda v.\mu !.[\omega] \lambda z.\mu !.\ulcorner R \lrcorner^{\mu}_{\mathrm{N}}\, v\, \mu\beta.\ulcorner Q \lrcorner^{\mu}_{\mathrm{N}}\, u \\
=_{\mathrm{N}} & \lambda u.\mu !.[\omega] \lambda z.\mu !.\ulcorner R \lrcorner^{\mu}_{\mathrm{N}}\, u\, \mu\beta.\ulcorner Q \lrcorner^{\mu}_{\mathrm{N}} \\
=_{\mathrm{V}} & \ulcorner P \lrcorner^{\mu'}_{\mathrm{N}}
\end{array}$$
□

Theorem 7.8 *For all $\mathcal{X}$-terms P, P', if $P \to_{\mathrm{N}}{}^{*} P'$, then $\llbracket P \rrbracket^{\Omega}_{\mathrm{N}} =_{\mathrm{N}} \llbracket P' \rrbracket^{\Omega}_{\mathrm{N}}$.*

7.2 Call by Value

We will now show we can achieve similar results for CBV-reduction.

Definition 7.9 (Call by value) For P any $\mathcal{X}$-term, we define $\llbracket P \rrbracket^{\mu}_{\mathrm{V}}$ by structural induction:

$$\begin{array}{rcl}
\llbracket \langle x{\cdot}\alpha \rangle \rrbracket^{\mu}_{\mathrm{V}} & \triangleq & [\alpha](x)^{\star} \\
\llbracket \widehat{y}P\widehat{\beta}{\cdot}\alpha \rrbracket^{\mu}_{\mathrm{V}} & \triangleq & [\alpha](\lambda y.\mu\beta.\,\llbracket P \rrbracket^{\mu}_{\mathrm{V}})^{\star} \\
\llbracket P\widehat{\beta}\,[x]\,\widehat{y}Q \rrbracket^{\mu}_{\mathrm{V}} & \triangleq & [\omega]\mu\beta.\,\llbracket P \rrbracket^{\mu}_{\mathrm{V}}\,\lambda v.\mu !.[\omega]x\,v\,\lambda y.\mu !.\,\llbracket Q \rrbracket^{\mu}_{\mathrm{V}} \\
\llbracket P\widehat{\alpha} \dagger \widehat{x}Q \rrbracket^{\mu}_{\mathrm{V}} \triangleq \llbracket P\widehat{\alpha} \nearrow \widehat{x}Q \rrbracket^{\mu}_{\mathrm{V}} & \triangleq & [\omega]\mu\alpha.\,\llbracket P \rrbracket^{\mu}_{\mathrm{V}}\,\lambda x.\mu !.\,\llbracket Q \rrbracket^{\mu}_{\mathrm{V}} \\
\llbracket P\widehat{\alpha} \nwarrow \widehat{x}Q \rrbracket^{\mu}_{\mathrm{V}} & \triangleq & [\omega]\lambda x.\mu !.\,\llbracket Q \rrbracket^{\mu}_{\mathrm{V}}\,\mu\alpha.\,\llbracket P \rrbracket^{\mu-}_{\mathrm{V}}
\end{array}$$

Contrary to the results obtained for the λ-calculus, we can achieve preservation of types via the CBV-interpretation using the type-interpretation defined above for the CBN-case.

Proposition 7.10 (Conservation of types in CBV) *If $P : \Delta \vdash_{\mathcal{X}} \Gamma$, then $\llbracket \Gamma \rrbracket^{\mu'}_{\mathrm{V}} \vdash_{\lambda\mu} \llbracket P \rrbracket^{\mu}_{\mathrm{V}} : cmd \mid \omega{:}\Omega,\, \llbracket \Delta \rrbracket^{\mu}_{\mathrm{V}}$.*

Proof: By induction on the structure of derivations:

(cap): Then $P \triangleq \langle x{\cdot}\alpha \rangle$, and there exists T such that $x{:}T$ and $\alpha{:}T$. As $x{:}\,\llbracket T \rrbracket^{\mu'}_{\mathrm{V}}$ and $\alpha{:}\,\llbracket T \rrbracket^{\mu}_{\mathrm{V}}$ in the translation, the command $[\alpha](x)^{\star}$ is well formed.

(exp): By induction

$$\llbracket \Gamma \rrbracket^{\mu'}_{\mathrm{V}}, y{:}\,\llbracket A \rrbracket^{\mu'}_{\mathrm{V}} \vdash_{\lambda\mu} \llbracket P \rrbracket^{\mu}_{\mathrm{V}} : cmd \mid \beta{:}\,\llbracket B \rrbracket^{\mu}_{\mathrm{V}}, \alpha{:}\,\llbracket A{\to}B \rrbracket^{\mu}_{\mathrm{V}}, \omega{:}\Omega,\, \llbracket \Delta \rrbracket^{\mu}_{\mathrm{V}}$$

As $\llbracket A{\to}B \rrbracket^{\mu'}_{\mathrm{V}} \triangleq \llbracket A \rrbracket^{\mu'}_{\mathrm{V}} \to \llbracket B \rrbracket^{\mu}_{\mathrm{V}}$, we get

$$\llbracket \Gamma \rrbracket^{\mu'}_{\mathrm{V}} \vdash_{\lambda\mu} \lambda y.\mu\beta.\,\llbracket P \rrbracket^{\mu}_{\mathrm{V}} : \llbracket A{\to}B \rrbracket^{\mu'}_{\mathrm{V}} \mid \alpha{:}\,\llbracket A{\to}B \rrbracket^{\mu}_{\mathrm{V}}, \omega{:}\Omega,\, \llbracket \Delta \rrbracket^{\mu}_{\mathrm{V}}$$

as expected and we conclude as for the previous case.

(med): By induction

$$\llbracket \Gamma \rrbracket^{\mu'}_{\mathrm{V}}, x{:}\,\llbracket A \rrbracket^{\mu'}_{\mathrm{V}} \to \llbracket B \rrbracket^{\mu}_{\mathrm{V}} \vdash_{\lambda\mu} \mu\beta.\,\llbracket P \rrbracket^{\mu}_{\mathrm{V}} : \llbracket A \rrbracket^{\mu}_{\mathrm{V}} \mid \omega{:}\Omega,\, \llbracket \Delta \rrbracket^{\mu}_{\mathrm{V}}$$

and

$$\llbracket \Gamma \rrbracket^{\mu'}_{\mathrm{V}}, x{:}\,\llbracket A \rrbracket^{\mu'}_{\mathrm{V}} \to \llbracket B \rrbracket^{\mu}_{\mathrm{V}} \vdash_{\lambda\mu} \lambda y.\mu !.\,\llbracket Q \rrbracket^{\mu}_{\mathrm{V}} : \llbracket B \rrbracket^{\mu'}_{\mathrm{V}} {\to} \Omega \mid \omega{:}\Omega,\, \llbracket \Delta \rrbracket^{\mu}_{\mathrm{V}}$$

Therefore we get

$$\llbracket \Gamma \rrbracket^{\mu'}_{\mathrm{V}}, x{:}\,\llbracket A \rrbracket^{\mu'}_{\mathrm{V}} \to \llbracket B \rrbracket^{\mu}_{\mathrm{V}} \vdash_{\lambda\mu} \lambda v.\mu !.[\omega]x\,v\,\lambda y.\mu !.\,\llbracket Q \rrbracket^{\mu}_{\mathrm{V}} : \llbracket A \rrbracket^{\mu'}_{\mathrm{V}} {\to} \Omega \mid \llbracket \Delta \rrbracket^{\mu}_{\mathrm{V}}$$

The result is clear.

$(left\ cut)$: By induction hypothesis

$$\llbracket \Gamma \rrbracket^{\mu}_{\mathrm{V}} \vdash_{\lambda\mu} \lambda x.\mu !.\,\llbracket Q \rrbracket^{\mu}_{\mathrm{V}} : \llbracket T \rrbracket^{\mu'}_{\mathrm{V}} {\to} \Omega \mid \omega{:}\Omega,\, \llbracket \Delta \rrbracket^{\mu}_{\mathrm{V}}$$

and

$$\llbracket \Gamma \rrbracket^{\mu}_{\mathrm{V}} \vdash_{\lambda\mu} \mu\alpha.\,\llbracket P \rrbracket^{\mu}_{\mathrm{V}} : \llbracket T \rrbracket^{\mu}_{\mathrm{V}} \mid \omega{:}\Omega,\, \llbracket \Delta \rrbracket^{\mu}_{\mathrm{V}}$$

Then

$$\llbracket \Gamma \rrbracket^{\mu}_{\mathrm{V}} \vdash_{\lambda\mu} \llbracket P\widehat{\alpha} \dagger \widehat{x}Q \rrbracket^{\mu}_{\mathrm{V}} : cmd \mid \omega{:}\Omega,\, \llbracket \Delta \rrbracket^{\mu}_{\mathrm{V}}$$

$(right\ cut)$: The induction hypotheses are the same. We know α is introduced in P, but whatever α, the application is well formed since $\mu\alpha.\,\llbracket P \rrbracket^{\mu-}_{\mathrm{V}} {:} [T]$. $\square$

Lemma 7.11 *If α is introduced in P and V any value, then*

$$\mu\alpha.\llbracket P \rrbracket_{\mathrm{V}}^{\mu} V \;=_{\mathrm{v}}\; V\,(\mu\alpha.\llbracket P \rrbracket_{\mathrm{V}}^{\mu})^{-}$$

Proof: By hypothesis, $\llbracket P \rrbracket_{\mathrm{V}}^{\mu} = \;\triangleq\; [\alpha](U)^{\star}$ where U is either a variable or an abstraction. Therefore

$$\begin{array}{rcl}
(\mu\alpha.\llbracket P \rrbracket_{\mathrm{V}}^{\mu})^{-} & \triangleq & \mu\tau.[\omega]\mu\alpha.[\alpha](U)^{\star}\,\lambda t.\mu!.[\tau]t \\
& =_{\mathrm{v}} & \mu\tau.[\omega]\mu\alpha.[\alpha](U)^{\star}\,\lambda t.\mu!.[\tau]t \\
& =_{\mathrm{v}} & \mu\tau.[\omega]\mu\alpha.[\alpha]\lambda t.\mu!.[\tau]t\,U \\
& =_{\mathrm{v}} & \mu\tau.[\omega]\mu\alpha.[\alpha]\mu!.[\tau]U \\
& =_{\mathrm{v}} & U
\end{array}$$

Then

$$\begin{array}{rcl}
\mu\alpha.[\alpha](U)^{\star}\,V & =_{\mathrm{v}} & \mu\alpha.[\alpha](U)^{\star}\,V \\
& =_{\mathrm{v}} & V\,U \\
& =_{\mathrm{v}} & V\,(\mu\alpha.\llbracket P \rrbracket_{\mathrm{V}}^{\mu})^{-} \qquad \square
\end{array}$$

The last result of this section is:

Theorem 7.12 *For all $\mathcal{X}$-terms P, P', if $P \rightarrow_{\mathrm{v}} P'$, then $\llbracket P \rrbracket_{\mathrm{V}}^{\Omega} =_{\mathrm{v}} \llbracket P' \rrbracket_{\mathrm{V}}^{\Omega}$.*

8 Main result

Using the results achieved above, we can now link provability in LK($\rightarrow$) typability in $\lambda\mu$, and formulate the main result of this paper:

Theorem 8.1 *Let S denotes any of the* CBN *or* CBV *strategy. For all circuits P in $\mathcal{X}$ such that $P : \Gamma \vdash_{\mathcal{X}} \Delta$, and for each type T occurring as $\tau{:}T$ in the conclusions Δ, there exists a $\lambda\mu$-term M_P such that $\Gamma \vdash_{\lambda\mu} M_P : T \mid \Delta \setminus \{\tau{:}T\}$. Moreover, if $P \rightarrow {}_{S}P'$ with respect to the strategy, then $M_P =_S M_{P'}$.*

References

[1] S. Bakel, S. Lengrand, and P. Lescanne. The language $\mathcal{X}$: circuits, computations and classical logic. In Mario Coppo, Elena Lodi, and G. Michele Pinna, editors, *Proceedings of Ninth Italian Conference on Theoretical Computer Science (ICTCS'05), Siena, Italy*, volume 3701 of *Lecture Notes in Computer Science*, pages 81–96. Springer-Verlag, 2005.

[2] S. Bakel and J. Raghunandan. Capture avoidance and garbage collection for $\mathcal{X}$. Submitted, 2006.

[3] H. Barendregt. *The Lambda Calculus: its Syntax and Semantics*. North-Holland, Amsterdam, revised edition, 1984.

[4] N. G. Bruijn. A namefree lambda calculus with facilities for internal definition of expressions and segments. TH-Report 78-WSK-03, Technological University Eindhoven, Netherlands, Department of Mathematics, 1978.

[5] Pierre-Louis Curien and Hugo Herbelin. The duality of computation. In *Proceedings of the 5 th ACM SIGPLAN International Conference on Functional Programming (ICFP'00)*, pages 233–243. ACM, 2000.

[6] G. Gentzen. Untersuchungen über das Logische Schliessen. *Mathematische Zeitschrift*, 39:176–210 and 405–431, 1935. English translation in [13], Pages 68–131.

[7] T. Griffin. A formulae-as-types notion of control. In *Proceedings of the 17th Annual ACM Symposium on Principles Of Programming Languages, Orlando (Fla., USA)*, pages 47–58, 1990.

[8] Ph. Groote. On the relation between the $\lambda\mu$-calculus and the syntactic theory of sequential control. In Springer-Verlag, editor, *Proceedings of the (th International Conference on Logic Programming and Automated Reasoning, (LPAR'94)*, volume 822 of *Lecture Notes in Computer Science*, pages 31–43, 1994.

[9] Jean-Louis Krivine. Classical logic, storage operators and second-order lambda-calculus. *Ann. Pure Appl. Logic*, 68(1):53–78, 1994.

[10] Stéphane Lengrand. Call-by-value, call-by-name, and strong normalization for the classical sequent calculus. In Bernhard Gramlich and Salvador Lucas, editors, *Electronic Notes in Theoretical Computer Science*, volume 86. Elsevier, 2003.

[11] C.-H. L. Ong and C. A. Stewart. A Curry-Howard foundation for functional computation with control. In *Proceedings of the 24th Annual ACM Symposium on Principles Of Programming Languages, Paris (France)*, pages 215–227, 1997.

[12] M. Parigot. An algorithmic interpretation of classical natural deduction. In *Proc. of Int. Conf. on Logic Programming and Automated Reasoning, LPAR'92*, volume 624 of *Lecture Notes in Computer Science*, pages 190–201. Springer-Verlag, 1992.

[13] M. E. Szabo, editor. *The Collected Papers of Gerhard Gentzen*. Studies in Logic and the Foundations of Mathematics. North-Holland, 1969.

[14] Christian Urban. *Classical Logic and Computation*. PhD thesis, University of Cambridge, October 2000.

[15] Philip Wadler. Call-by-Value is Dual to Call-by-Name. In *Proceedings of the eighth ACM SIGPLAN international conference on Functional programming*, pages 189 – 201, 2003.

[16] A N. Whitehead and B. Russell. *Principia Mathematica*. Cambridge University Press, 2nd edition, 1925.