Laboratoire de l'Informatique du Parallélisme

École Normale Supérieure de Lyon
Unité Mixte de Recherche CNRS-INRIA-ENS LYON-UCBL
n° 5668

The language $\mathcal{X}$: circuits, computations and Classical Logic

Steffen van Bakel
St´ephane Lengrand
Pierre Lescanne

March 2005

Research Report N° RR2005-11

École Normale Supérieure de Lyon
46 Allée d'Italie, 69364 Lyon Cedex 07, France
Téléphone : +33(0)4.72.72.80.37
Télécopieur : +33(0)4.72.72.80.80
Adresse électronique : lip@ens-lyon.fr

The language $\mathcal{X}$: circuits, computations and Classical Logic

Steffen van Bakel
St´ephane Lengrand
Pierre Lescanne

March 2005

Abstract

$\mathcal{X}$ is an untyped language for describing circuits by composition of basic components. This language is well suited to describe structures which we call "circuits" and which are made of parts that are connected by wires. Moreover $\mathcal{X}$ gives an expressive platform on which algebraic objects and many different (applicative) programming paradigms can be mapped. In this paper we will present the syntax and reduction rules for $\mathcal{X}$ and some its potential uses.
To demonstrate the expressive power of $\mathcal{X}$, we will show how, even in an untyped setting, elaborate calculi can be embedded, like the naturals, the λ-calculus, Bloe and Rose's calculus of explicit substitutions $\lambda\mathbf{x}$, Parigot's $\lambda\mu$ and Curien and Herbelin's $\lambda\mu\tilde{\mu}$.

Keywords: Language design, mobility, circuits, classical logic, Curry-Howard correspondance

Résumé

$\mathcal{X}$ est un langage non typ´e conçu pour d´ecrire les circuits par composition de «briques» de base. Ce langage s'adapte parfaitement à la description des structures que nous appelons «circuits» et qui sont faites de composants connect´es par des fils. De plus, $\mathcal{X}$ fournit une plate-forme expressive sur laquelle des objets alg´ebriques et de nombreux paradigmes de programmation (applicative) de toutes sortes peuvent être appliqu´es. Dans ce rapport, nous pr´esenterons la syntaxe de $\mathcal{X}$, ses règles de r´eduction et certaines de ses utilisations potentielles.
Pour mettre en lumière le pouvoir expressif de $\mathcal{X}$, nous montrerons comment, même dans un cadre non typ´e, on peut y plonger des calculs relativement sophistiqu´es, comme les entiers naturels, le λ-calcul, le calcul de substitutions explicites $\lambda\mathbf{x}$ de Bloe et Rose, le calcul $\lambda\mu$ de Parigot et le calcul $\lambda\mu\tilde{\mu}$ de Curien et Herbelin.

Mots-clés: Conception de langage, mobilit´e, circuits, logique classique, correspondance de Curry-Howard

1 Introduction

In this paper we will present a language called $\mathcal{X}$ that describes circuits, and its reduction rules that couple (weld) circuits. Among others, we will show how $\mathcal{X}$ can be used to describe the behaviour of functional programming languages at a very low level of granularity.

$\mathcal{X}$ as a language for describing circuits

The basic pieces of $\mathcal{X}$ can be understood as components with entrance and exit wires and ways to describe how to connect them to build larger circuits. Those component will be quickly surveyed in the introduction and receive a more detailed treatment in Section 2. We call *"circuits"* the structures we build, because they are made of components connected by wires.

$\mathcal{X}$ as a syntax for the sequent calculus

Starting from the proof of Dragalin [13], Herbelin proposed in his PhD [17] a Curry-Howard correspondence; this was more elaborated in [10] leading to the definition of the language $\lambda\mu\tilde{\mu}$ (see Section 6). Among other approaches we need to mention [16, 11, 12, 6]; more generally, this work has connections with linear logic [15]. The relation between call-by-name and call-by-value in the context of $\lambda\mu\tilde{\mu}$ is studied in detail in [27].

The origins of the language $\mathcal{X}$ we propose in this paper lie in the notations for the sequent calculus as presented first by Urban in his PhD thesis [25] and later studied in relation with $\lambda\mu\tilde{\mu}$ [10] by Lengrand [19]. The study of the connections between $\mathcal{X}$ and $\lambda\mu\tilde{\mu}$ as reported on in [10] was not fortuitous, and was taken as starting point for our investigations; in this paper, we will show in detail the interplay between $\mathcal{X}$ and $\lambda\mu\tilde{\mu}$.

An interesting aspect of $\mathcal{X}$ is its role within the context of cut-elimination. For this, $\mathcal{X}$ is naturally typed. One main step forward with respect to previous work is that we went to an untyped language which serves as a expressive framework for representing the untyped lambda calculus, the untyped calculus of explicit substitutions and the untyped language $\lambda\mu\tilde{\mu}$.

Some of the contributions of this paper are to make the notation more intuitive and readable by moving to an infix notation, and to insist on the computational aspect. This is achieved by studying $\mathcal{X}$ in the context of the normal functional programming languages paradigms, but, more importantly, to cut the link between $\mathcal{X}$ and Classical Logic. We achieve this by studying our language *without types*; this way, we also consider circuits that do *not* correspond to proofs. In particular, we consider also non-termination circuits. In fact, we aim to study $\mathcal{X}$ outside the context of Classical Logic in much the same way as the λ-calculus is studied outside the context of Intuitionistic Logic.

$\mathcal{X}$ as a fine grained operational model of computation

When taking the λ-calculus as a model for programming languages, the operational behaviour is provided by *β-contraction*. As is well known, β-contraction expresses how to calculate the value of a function applied to a parameter. In this, the parameter is used to instantiate occurrences of the bound variable in the body via the process of *substitution*. This description is rather basic as it says nothing on the actual *cost* of the

substitution, which is quite high at run-time. Usually, a calculus of *explicit substitutions* [7, 1, 21, 20] is considered better suited for an accurate account of the substitution process and its implementation. When we refer to the calculus of explicit substitution we rather intend $\lambda\mathbf{x}$, the calculus of *explicit substitution with explicit names*, due to Bloo and Rose [7]. $\lambda\mathbf{x}$ gives a better account of substitution as it integrates substitutions as first class citizens, decomposes the process of inserting a term into atomic actions, and explains in detail how substitutions are distributed through terms to be eventually evaluated at the variable level.

In this paper, we will show that the level of description reached by explicit substitutions can in fact be lowered substantially. In $\mathcal{X}$, we reach a 'subatomic' level by decomposing explicit substitutions into smaller components. At this level, the calculus $\mathcal{X}$ explains how substitutions and terms interact.

The calculus is actually symmetric [4] and, unlike $\lambda\mathbf{x}$ where a substitution is applied to a term, a term in $\mathcal{X}$ can also be applied to a substitution. Their interaction percolates subtly and gently through the term or substitution according to the direction that has been chosen. We will see that the these two kinds of interaction have a direct connection with call-by-value and call-by-name strategies, that both have a natural description in $\mathcal{X}$.

The ingredients of the syntax

It is important to note that $\mathcal{X}$ does *not* have variables[1] –like λ-calculus or $\lambda\mu\tilde{\mu}$– as possible places where terms might be inserted; instead, $\mathcal{X}$ has *wires*, also called *connectors*, that can occur free or bound in a term. As for the λ-calculus, the binding of a wire indicates that it is *active* in the computation; other than in the λ-calculus, however, the binding is not part of a term that is involved in the interaction, but is part of the interaction itself.

There are two kinds of wires: *sockets* that are reminiscent of values and *plugs* (corresponding to *variables* and *covariables*, respectively, in [27]) that are reminiscent of continuations. Wires are not supposed to denote a location in a term like variables in λ-calculus. Rather, wires are seen a bit like *ropes* that can be *knotted* or *tightened* (like *chemical bonds*) with ropes of other components.

This, in fact, corresponds in a way to the practice of sailing. Sailors give a very specific name to each rope (*main sail halyard*, *port jib sheet*, etc.), and on a modern competition sailboat every rope has its own colour to be sure that one tightens (or loosens) a rope to (or from) its appropriate place or with (or from) the appropriate rope; loosening the wrong rope can be catastrophic. In $\mathcal{X}$, those colours naturally become *types*, and like a rope has a colour, a wire has a type.

One specificity of $\mathcal{X}$ is that syntactic constructors bind *two* wires, one of each kind. In $\mathcal{X}$, bound wires receive a hat, so to show that x is bound we write $\hat{x}$ [28, 29]. That a wire is bound in a term implies, naturally, that this wire is unknown outside that term, but also that it 'interacts' with another 'opposite' wire that is bound into another term. The interaction differs from one constructor to another, and is ruled by basic reductions (see Section 2). In addition to bound wires an introduction rule exhibits a free wire, that is exposed and connectable. Often –but not always– this exhibition corresponds to the creation of the wire.

[1] We encourage the reader to not become confused by the use of names like x for the class of connectors that are called plugs; these names are, in fact, inherited from $\lambda\mu\tilde{\mu}$.

Contents of this paper

In this paper we will present the formal definitions for $\mathcal{X}$, via syntax and reduction rules, and will show that the system is well-behaved by stating a number of essential properties. We will define a notion of simple type assignment for terms in $\mathcal{X}$, in that we will define a system of derivable judgements for which the terms of $\mathcal{X}$ are witnesses; we will show a soundness result for this system by showing that a subject-reduction result holds.

We will also compare $\mathcal{X}$ with a number of its predecessors. In fact, we will show that a number of well-know calculi are easily, elegantly and surprisingly effectively implementable in $\mathcal{X}$. For anyone familiar with the problem of expressibility, in view of the fact that $\mathcal{X}$ is substitution-free, these result are truly novel. With the exception of the calculus $\lambda\mu\tilde{\mu}$, the converse is unobtainable. This can easily be understood from the fact that the vast majority of calculi in our area is confluent (Church-Rosser), whereas $\mathcal{X}$ is not.

2 The $\mathcal{X}$-calculus

The circuits that are the objects of $\mathcal{X}$ are built with three kinds of building stones, or constructors, called *capsule*, *export* and *mediator*.. In addition there is an operator we call *weld*, which is handy for describing circuit construction, and which will be eliminated eventually by *rules*. In addition we give *congruence among circuits*.

2.1 The operators

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 by an operator, say $\oplus$, 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 $P\hat{x}\oplus\hat{\alpha}Q$ the name x is not reachable outside P and the name α is not reachable outside Q. This reminds of construction like $\forall x.P$ or $\lambda x.M$ in logic where the name x is not known outside the expressions. Logicians say that those names are *bound*. Likewise, in $\mathcal{X}$, in a construction like $P\hat{x}\oplus\hat{\alpha}Q$ where α is a plug of P and x is a socket of Q, there are *two bound names* namely α and x, that are bound in the interaction. We use the *"hat"*-notation, keeping in line with the old tradition of *Principia Mathematica* [28], writing $\hat{x}$ to say that x is bound. For connecting P and Q through $\oplus$ we write $P\hat{\alpha}\oplus\hat{x}Q$.

Definition 2.1 (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*.*

$$P, Q ::= \langle x.\alpha\rangle \mid \widehat{y}P\widehat{\beta}\cdot\alpha \mid P\widehat{\beta}\,[y]\,\widehat{x}Q \mid P\widehat{\alpha}\dagger\widehat{x}Q$$

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 2.2 and Barendregt's Convention (see also below).

Assume that the terms of the left-hand sides of the rules ***introduce the socket*** x ***and the plug*** α.

$$
\begin{array}{lr}
(\textit{cap-cap}): & \langle y.\alpha\rangle\widehat{\alpha}\dagger\widehat{x}\langle x.\beta\rangle \rightarrow \langle y.\beta\rangle \\
(\textit{exp}): & (\widehat{y}P\widehat{\beta}\cdot\alpha)\widehat{\alpha}\dagger\widehat{x}\langle x.\gamma\rangle \rightarrow \widehat{y}P\widehat{\beta}\cdot\gamma \\
(\textit{med}): & \langle y.\alpha\rangle\widehat{\alpha}\dagger\widehat{x}(P\widehat{\beta}\,[x]\,\widehat{z}Q) \rightarrow P\widehat{\beta}\,[y]\,\widehat{z}Q \\
(\textit{exp-med}): & (\widehat{y}P\widehat{\beta}\cdot\alpha)\widehat{\alpha}\dagger\widehat{x}(Q\widehat{\gamma}\,[x]\,\widehat{z}R) \rightarrow Q\widehat{\gamma}\dagger\widehat{y}P\widehat{\beta}\dagger\widehat{z}R
\end{array}
$$

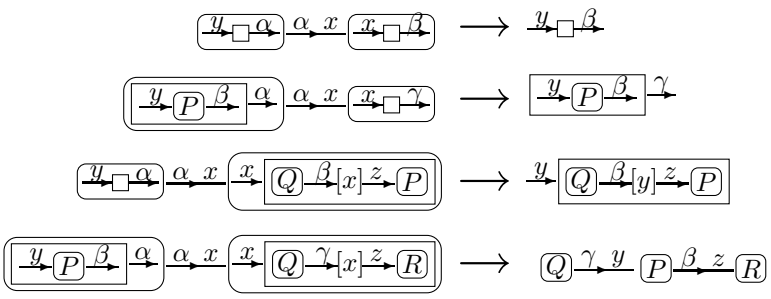


Figure 1: The logical rules and their diagrammatical representation

We could justify the constructions of $\mathcal{X}$ through the example of the "translations" in a huge international organisation[2]. The arguments below will be better established and formalised in Section 5 when we will speak about *types* which are the correct framework. But of course $\mathcal{X}$ is basically an untyped language.

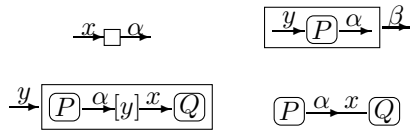
Suppose wires carry words in a language like Estonian, or Portuguese, etc, but also translators from language to language like "French to Dutch." A *capsule* $\langle x.\alpha\rangle$ connects inside the socket x with the plug α. Everything entering the capsule on x in one language will leave it in the same language on α. An *export* $(\widehat{y}P\widehat{\alpha}\cdot\beta)$ can be seen as follows: P provides a device that transforms words in a language received on y to words in a(nother) language returned on β, therefore $(\widehat{y}P\widehat{\beta}\cdot\alpha)$ is a translator that is returned on a specific wire α which can be connected later on. An export can also be seen as *T-diagrams* like those used in compiling technology for bootstrapping (see [2] Section 11.2). A *mediator* $(P\widehat{\alpha}\,[y]\,\widehat{x}Q)$ is required when one tries to connect two wires α and x to carry words from different languages. To be able to achieve that task, one needs to receive a translator to be connected to the wire y, a socket.

The operator $\dagger$ is called a *weld* and a term of the form $(P\widehat{\alpha}\dagger\widehat{x}Q)$ is called a *welt term*, and corresponds to an operation on the *switch board*. A weld is specific, it just connects (more precisely, it *welds*) two circuits, connecting the socket x of Q to the plug α of P; this assumes that the language expected on x agrees with the language delivered by α. The weld expresses the need for a rewiring of the switch board: a language is on offer on a plug, and demanded on a socket, and "dealing" with the weld, which expresses the need for the connection to be established, will cause the weld to be eventually eliminated.

The calculus, defined by the reduction rules (Section 2.2) explains in detail how welds are distributed through circuits to be eventually erased at the level of capsules.

[2]see `http://europa.eu.int/comm/translation/index_en.htm`)

Diagrammatically, we represent the basic circuits as:



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

Definition 2.2 *The* free sockets *and* free plugs *in a circuit are:*

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

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

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

We will normally adopt Barendregt's convention (called convention on variables by Barendregt, but here it will be a convention on names). An exception to that convention is the definition of natural numbers in Section 3.

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., $\hat{y}P[y/x]$ and this renaming can be internalise (see Section 2.5).

Definition 2.3 (Renaming of sockets and plugs)

$$\begin{aligned}
\langle x.\alpha\rangle[y/x] &= \langle y.\alpha\rangle\\
\langle u.\alpha\rangle[y/x] &= \langle u.\alpha\rangle, && x\neq u\\
(\widehat{z}P\widehat{\beta}\cdot\alpha)[y/x] &= \widehat{z}(P[y/x])\widehat{\beta}\cdot\alpha\\
(P\widehat{\alpha}\,[x]\,\widehat{z}Q)[y/x] &= (P[y/x])\widehat{\alpha}\,[y]\,\widehat{z}(Q[y/x])\\
(P\widehat{\alpha}\,[u]\,\widehat{z}Q)[y/x] &= (P[y/x])\widehat{\alpha}\,[u]\,\widehat{z}(Q[y/x]), && x\neq u\\
(P\widehat{\alpha}\dagger\widehat{z}Q)[y/x] &= (P[y/x])\widehat{\alpha}\dagger\widehat{z}(Q[y/x])
\end{aligned}$$

$$\begin{aligned}
\langle x.\alpha\rangle[\beta/\alpha] &= \langle x.\beta\rangle\\
\langle x.\gamma\rangle[\beta/\alpha] &= \langle x.\gamma\rangle, && \alpha\neq\gamma\\
(\widehat{z}P\widehat{\delta}\cdot\alpha)[\beta/\alpha] &= \widehat{z}(P[\beta/\alpha])\widehat{\delta}\cdot\beta\\
(\widehat{z}P\widehat{\delta}\cdot\gamma)[\beta/\alpha] &= \widehat{z}(P[\beta/\alpha])\widehat{\delta}\cdot\gamma, && \alpha\neq\gamma\\
(P\widehat{\delta}\,[x]\,\widehat{z}Q)[\beta/\alpha] &= (P[\beta/\alpha])\widehat{\delta}\,[y]\,\widehat{z}(Q[\beta/\alpha])\\
(P\widehat{\delta}\dagger\widehat{z}Q)[\beta/\alpha] &= (P[\beta/\alpha])\widehat{\delta}\dagger\widehat{z}(Q[\beta/\alpha])
\end{aligned}$$

Left propagation

$$
\begin{array}{rrll}
(d\text{L}): & \langle y.\alpha\rangle\widehat{\alpha}\nleftarrow\widehat{x}P & \to \langle y.\alpha\rangle\widehat{\alpha}\dagger\widehat{x}P & \\
(\nleftarrow \textit{cap}): & \langle y.\beta\rangle\widehat{\alpha}\nleftarrow\widehat{x}P & \to \langle y.\beta\rangle, & \beta \neq \alpha \\
(\nleftarrow \textit{exp1}): & (\widehat{y}Q\widehat{\beta}\cdot\alpha)\widehat{\alpha}\nleftarrow\widehat{x}P & \to (\widehat{y}(Q\widehat{\alpha}\nleftarrow\widehat{x}P)\widehat{\beta}\cdot\gamma)\widehat{\gamma}\dagger\widehat{x}P, & \gamma \textit{ fresh} \\
(\nleftarrow \textit{exp2}): & (\widehat{y}Q\widehat{\beta}\cdot\gamma)\widehat{\alpha}\nleftarrow\widehat{x}P & \to \widehat{y}(Q\widehat{\alpha}\nleftarrow\widehat{x}P)\widehat{\beta}\cdot\gamma, & \gamma \neq \alpha \\
(\nleftarrow \textit{med}): & (Q\widehat{\beta}\,[z]\,\widehat{y}R)\widehat{\alpha}\nleftarrow\widehat{x}P & \to (Q\widehat{\alpha}\nleftarrow\widehat{x}P)\widehat{\beta}\,[z]\,\widehat{y}(R\widehat{\alpha}\nleftarrow\widehat{x}P) & \\
(\nleftarrow \textit{weld}): & (Q\widehat{\beta}\dagger\widehat{y}R)\widehat{\alpha}\nleftarrow\widehat{x}P & \to (Q\widehat{\alpha}\nleftarrow\widehat{x}P)\widehat{\beta}\dagger\widehat{y}(R\widehat{\alpha}\nleftarrow\widehat{x}P) &
\end{array}
$$

Right propagation

$$
\begin{array}{rrll}
(d\text{R}): & P\widehat{\alpha}\nrightarrow\widehat{x}\langle x.\beta\rangle & \to P\widehat{\alpha}\dagger\widehat{x}\langle x.\beta\rangle & \\
(\nrightarrow \textit{cap}): & P\widehat{\alpha}\nrightarrow\widehat{x}\langle y.\beta\rangle & \to \langle y.\beta\rangle, & y \neq x \\
(\nrightarrow \textit{exp}): & P\widehat{\alpha}\nrightarrow\widehat{x}(\widehat{y}Q\widehat{\beta}\cdot\gamma) & \to \widehat{y}(P\widehat{\alpha}\nrightarrow\widehat{x}Q)\widehat{\beta}\cdot\gamma & \\
(\nrightarrow \textit{med1}): & P\widehat{\alpha}\nrightarrow\widehat{x}(Q\widehat{\beta}\,[x]\,\widehat{y}R) & \to P\widehat{\alpha}\dagger\widehat{z}((P\widehat{\alpha}\nrightarrow\widehat{x}Q)\widehat{\beta}\,[z]\,\widehat{y}(P\widehat{\alpha}\nrightarrow\widehat{x}R)), & z \textit{ fresh} \\
(\nrightarrow \textit{med2}): & P\widehat{\alpha}\nrightarrow\widehat{x}(Q\widehat{\beta}\,[z]\,\widehat{y}R) & \to (P\widehat{\alpha}\nrightarrow\widehat{x}Q)\widehat{\beta}\,[z]\,\widehat{y}(P\widehat{\alpha}\nrightarrow\widehat{x}R), & z \neq x \\
(\nrightarrow \textit{weld}): & P\widehat{\alpha}\nrightarrow\widehat{x}(Q\widehat{\beta}\dagger\widehat{y}R) & \to (P\widehat{\alpha}\nrightarrow\widehat{x}Q)\widehat{\beta}\dagger\widehat{y}(P\widehat{\alpha}\nrightarrow\widehat{x}R) &
\end{array}
$$

Figure 2: The propagation rules.

Renaming will play an important part in dealing with α-conversion, a problem we will discuss in Subsection 2.5.

2.2 The rules

It is important to know when a socket or a plug is introduced, i.e. is connectable, i.e. is exposed and unique. Indeed circuits that introduce sockets and plugs will play an important role in the reduction rules. 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. We say now formally what it means for a terms to *introduce* a socket or a plug (Urban [25] uses the terminology "freshly introduce").

Definition 2.4 (Introduction)

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

P **introduces** α**:** $P = \langle y.\alpha\rangle$ *or* $P = \widehat{x}Q\widehat{\beta}\cdot\alpha$ *with* $\alpha \notin \mathit{fp}(Q)$.

We first present a simple family of reduction rules. They say how to reduce a circuit that welds subcircuits that introduce connectors. They are naturally divided in four categories whenever a capsule is welt with a capsule, an export with a capsule, a capsule with a mediator or an export with a mediator. There is no other pattern in which a plug is introduced on the left of a $\dagger$ and a socket is introduced on the right.

Definition 2.5 (Logical Reduction) *The logical rules and their diagrammatical representation are given in Figure 1.*

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

Still in rule (*exp-med*) the reader may have noticed that we did not put parenthesis in the expression $Q\widehat{\gamma}\dagger\widehat{y}P\widehat{\beta}\dagger\widehat{z}R$, which therefore is officially not a circuit. Instead, we should have given both the circuits $(Q\widehat{\gamma}\dagger\widehat{y}P)\widehat{\beta}\dagger\widehat{z}R$ and $Q\widehat{\gamma}\dagger\widehat{y}(P\widehat{\beta}\dagger\widehat{z}R)$ as result of the rewriting. However there is, in fact, a kind of associativity at play which means that we can omit the parenthesis; this will be made more clear in the next section.

We now need to define how to reduce a welt 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 *bent* welds:

$$P ::= \ldots \mid P\widehat{\alpha}\nearrow\widehat{x}Q \mid P\widehat{\alpha}\nwarrow\widehat{x}Q$$

Circuits where welds are not bent are called *pure* (the diagrammatical representation of bent welds is the same as that for straightened out (i.e. not bent) welds). Bent welds are propagated through the terms.

Definition 2.6 (Bending the welds)

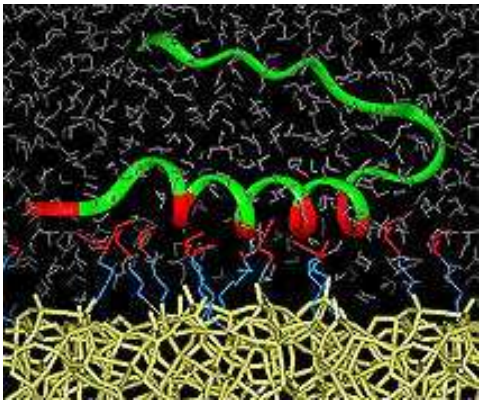
$$(\textit{bend-L}) : P\widehat{\alpha}\dagger\widehat{x}Q \rightarrow P\widehat{\alpha}\nearrow\widehat{x}Q, \textit{ if } P \textit{ does not introduce } \alpha$$
$$(\textit{bend-R}) : P\widehat{\alpha}\dagger\widehat{x}Q \rightarrow P\widehat{\alpha}\nwarrow\widehat{x}Q, \textit{ if } N \textit{ does not introduce } x$$

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 bent weld through sub-circuits. The direction of the bending shows in which direction the weld should be propagated, hence the two sets of reduction rules.

Definition 2.7 (Propagation Reduction) *The rules of propagation are given in Figure 2.*

We imagine the diffusion of bents like the diffusion of a protein at the level of molecular interaction (picture taken from [18]):



$$
\begin{array}{rlcll}
(\dagger\textit{-assoc}) & (P\widehat{\alpha}\dagger\widehat{x}Q)\widehat{\beta}\dagger\widehat{y}R & \overset{\mathcal{X}}{=} & P\widehat{\alpha}\dagger\widehat{x}(Q\widehat{\beta}\dagger\widehat{y}R) & \text{if } \beta\notin \mathit{fp}(P)\wedge x\notin \mathit{fs}(R)\\
(\textit{left-comm}) & P\widehat{\alpha}\dagger\widehat{x}(Q\widehat{\beta}\dagger\widehat{y}R) & \overset{\mathcal{X}}{=} & Q\widehat{\beta}\dagger\widehat{y}(P\widehat{\alpha}\dagger\widehat{x}R) & \text{if } x\notin \mathit{fs}(Q)\wedge y\notin \mathit{fs}(P)\\
(\textit{right-comm}) & (P\widehat{\alpha}\dagger\widehat{x}Q)\widehat{\beta}\dagger\widehat{y}R & \overset{\mathcal{X}}{=} & (P\widehat{\beta}\dagger\widehat{y}R)\widehat{\alpha}\dagger\widehat{x}Q & \text{if } \alpha\notin \mathit{fp}(R)\wedge\beta\notin \mathit{fp}(Q)
\end{array}
$$

Figure 3: The structural congruences for †

$$
(\textit{med-assoc})\ P\widehat{\alpha}\,[z]\,\widehat{x}(Q\widehat{\beta}\,[u]\,\widehat{y}R) \ \overset{\mathcal{X}}{=}\ (P\widehat{\alpha}\,[z]\,\widehat{x}Q)\widehat{\beta}\,[u]\,\widehat{y}R \qquad \text{if } \beta\notin \mathit{fp}(P)\backslash\{\alpha\}\ \wedge\ x\notin \mathit{fs}(R)\backslash\{y\}
$$

Figure 4: Structural congruence for the mediators

We will subscript the arrow that represents our reduction to indicate certain subsystems, defined by a reduction strategy: for example, we will write $\rightarrow_{\mathrm{A}}$ for the reduction that uses only rules in *Left propagation* or *Right propagation*. In fact, $\rightarrow_{\mathrm{A}}$ is the reduction that pushes $\not\leftarrow$ and $\not\backslash$ inward.

The rules $(\not\leftarrow \textit{exp}1)$ and $(\not\backslash \textit{med}1)$ deserve some attention. For instance, in the left-hand side of $(\not\leftarrow \textit{exp}1)$, 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 namely γ. Note that the weld associated with that γ is then unbent; this is because, after the bent welt has been pushed through $\widehat{y}(Q\widehat{\alpha}\not\leftarrow\widehat{x}P)\widehat{\beta}\cdot\gamma$ (so leaves a circuit with no bent weld), 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. So, in any case, it would be wrong to active the weld before the result of $Q\widehat{\alpha}\not\leftarrow\widehat{x}P$ (i.e. R) is known.

The same thing happens with x in $(\not\backslash \textit{med}1)$ and a new name z is created and the external weld is unbent.

2.3 Structural congruences

One describes the simplification theory of Abelian groups with two reduction rules, namely

$$
\begin{aligned}
x+0 &\rightarrow x\\
x+(-x) &\rightarrow 0
\end{aligned}
$$

and two congruences namely

$$
\begin{aligned}
x+(y+z) &= (x+y)+z\\
x+y &= y+x.
\end{aligned}
$$

The same kind of division in presentation appears in the π-calculus [22] under the name of *structural congruences*. Rewriting modulo a set of equations is the basis of rewriting logic [8, 9].

We observe the same in $\mathcal{X}$. We define two congruences for † that look like associativity and commutativity. (Actually, purists would say *left commutativity* $x+(y+z)=y+(x+z)$ and *right commutativity* $(x+y)+z=(x+z)+y$.)

Definition 2.8 *The structural congruences are given in Figure 3.*

Rule (†-*assoc*) allows us to write $P\widehat{\alpha}\dagger\widehat{x}Q\widehat{\beta}\dagger\widehat{y}R$ (provided the side-condition is fulfilled) since the order of the applications of † is irrelevant. Notice that the side-condition for this rule is the one we have indicated for (*exp-med*), and comes from the variable convention. This is consistent with the parenthesis-free notation we have used for the right-hand side of (*exp-med*). Notice that now writing $P\widehat{\alpha}\dagger\widehat{x}Q\widehat{\alpha}\dagger\widehat{x}R$ is licit. The second rule is left-commutativity and the third rule is right-commutativity.

There is another rule asserting the associativity of the mediators given in Figure 4. One may see that the side-condition is the same as for the weld, which is not surprising. We will freely write $P\widehat{\alpha}\ [z]\ \widehat{x}Q\widehat{\beta}\ [u]\ \widehat{y}R$.

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 [27], where, as in $\lambda\mu\tilde{\mu}$, only one notion of reduction is defined; the CBN-CBV result there was obtained via different interpretation functions from CBN/CBV calculi (see also Section 8).

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 (*bend*-L) and (*bend*-R), can both be fired.

The *critical pair* $\langle P\widehat{\alpha}\not\swarrow\widehat{x}Q, P\widehat{\alpha}\not\searrow\widehat{x}Q\rangle$. 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 bending whenever the above critical pair occurs:

Definition 2.9

- *The* CBV *strategy only bends a weld via* (*bend*-L) *when it could be bent in two ways; we write* $P \rightarrow_{\text{V}} Q$ *in that case.*
- *The* CBN *strategy only bends such a weld via* (*bend*-R)*; like above, we write* $P \rightarrow_{\text{N}} Q$.

We will now show some basic properties, which essentially show that the calculus is well-behaved.

Lemma 2.10 (Cancellation)

1. $P\widehat{\alpha}\not\swarrow\widehat{x}Q \rightarrow_{\text{A}} P$ *if* $\alpha \notin fp(P)$ *and* P *is pure.*
2. $P\widehat{\alpha}\dagger\widehat{x}Q \rightarrow_{\text{V}} P$ *if* $\alpha \notin fp(P)$ *and* P *is pure.*
3. $P\widehat{\alpha}\not\searrow\widehat{x}Q \rightarrow_{\text{A}} Q$ *if* $x \notin fs(Q)$ *and* Q *is pure.*
4. $P\widehat{\alpha}\dagger\widehat{x}Q \rightarrow_{\text{N}} Q$ *if* $x \notin fs(Q)$ *and* Q *is pure.*

We will now show that a weld with a capsule leads to renaming. First we show this, with the capsule on the right, for left welds in part 1, and generalise this to unlabelled welds in part 2. Similarly, we show this for right welds with the capsule on the left in part 3, and for unlabelled welds in part 4.

Lemma 2.11 (Renaming)

1. $P\widehat{\delta} \not\leftarrow \widehat{z}\langle z.\alpha\rangle \to_{\mathrm{A}} P[\alpha/\delta]$, *if* P *is pure.*
2. $P\widehat{\delta} \dagger \widehat{z}\langle z.\alpha\rangle \to P[\alpha/\delta]$, *if* P *is pure.*
3. $\langle z.\alpha\rangle\widehat{\alpha} \not\rightarrow \widehat{x}P \to_{\mathrm{A}} P[z/x]$, *if* P *is pure.*
4. $\langle z.\alpha\rangle\widehat{\alpha} \dagger \widehat{x}P \to P[z/x]$, *if* P *is pure.*

These results motivate the extension (in both strategies) of the reduction rules, formulating new rules in the shape of the above results.

2.5 α-conversion

Normally, renaming is an essential part of α-conversion, the process of renaming bound objects in a language to avoid clashes during computation. The most familiar context in which this occurs is of course the λ-calculus, where, when reducing a term like $(\lambda xy.xy)(\lambda xy.xy)$, α-conversion is essential. We will now discuss briefly the solution of [26], that deals accurately with this problem in $\mathcal{X}$.

Example 2.12 *Take the following reduction:*

$$\begin{gathered}(\widehat{y}\langle y.\rho\rangle\widehat{\rho}\cdot\gamma)\widehat{\gamma} \dagger \widehat{x}(\langle x.\delta\rangle\widehat{\delta}\,[x]\ \widehat{w}\langle w.\alpha\rangle) \to \\ (\textit{bend-}\mathrm{R}), (\not\leftarrow \textit{med}1), (d\mathrm{R}), (\textit{exp}) \\ (\widehat{y}\langle y.\rho\rangle\widehat{\rho}\cdot\gamma)\widehat{\gamma} \dagger \widehat{z}((\widehat{y}\langle y.\rho\rangle\widehat{\rho}\cdot\delta)\widehat{\delta}\,[z]\ \widehat{w}\langle w.\alpha\rangle) \to (\textit{exp-med}) \\ (\widehat{y}\langle y.\rho\rangle\widehat{\rho}\cdot\delta)\widehat{\delta} \dagger \widehat{y}\langle y.\rho\rangle\widehat{\rho} \dagger \widehat{w}\langle w.\alpha\rangle\end{gathered}$$

If we now re-introduce the omitted brackets, it is clear that we are in breach with Barendregt's convention: ρ *is both free and bound in* $(\widehat{y}\langle y.\rho\rangle\widehat{\rho}\cdot\delta)\widehat{\delta} \dagger \widehat{y}\langle y.\rho\rangle$*. If we were to continue the reduction, we obtain:*

$$\begin{gathered}((\widehat{y}\langle y.\rho\rangle\widehat{\rho}\cdot\delta)\widehat{\delta} \dagger \widehat{y}\langle y.\rho\rangle)\widehat{\rho} \dagger \widehat{w}\langle w.\alpha\rangle \to (\textit{exp}) \\ (\widehat{y}\langle y.\rho\rangle\widehat{\rho}\cdot\rho)\widehat{\rho} \dagger \widehat{w}\langle w.\alpha\rangle\end{gathered}$$

Notice that ρ *is not introduced in* $\widehat{y}\langle y.\rho\rangle\widehat{\rho}\cdot\rho$*, since* $\rho \in \mathit{fp}(\langle y.\rho\rangle)$*. So the weld is propagated, and we obtain:*

$$\begin{gathered}\to (\textit{bend-}\mathrm{L}), (\not\rightarrow \textit{exp}1), (d\mathrm{L}) \\ (\widehat{y}(\langle y.\rho\rangle\widehat{\rho} \dagger \widehat{w}\langle w.\alpha\rangle)\widehat{\rho}\cdot\rho)\widehat{\rho} \dagger \widehat{w}\langle w.\alpha\rangle \to (\textit{cap-cap}), (\textit{exp}) \\ \widehat{y}\langle y.\alpha\rangle\widehat{\rho}\cdot\alpha.\end{gathered}$$

This is not correct; by α*-conversion,* $(\widehat{y}\langle y.\rho\rangle\widehat{\rho}\cdot\rho)\widehat{\rho} \dagger \widehat{w}\langle w.\alpha\rangle$ *is actually* $(\widehat{y}\langle y.\sigma\rangle\widehat{\sigma}\cdot\rho)\widehat{\rho} \dagger \widehat{w}\langle w.\alpha\rangle$*, where the* ρ *is introduced, and we should have obtained* $\widehat{y}\langle y.\rho\rangle\widehat{\rho}\cdot\alpha$*.*

It is clear from this example that α-conversion is needed to some extent in any implementation of $\mathcal{X}$. The solution to this problem as proposed in [26] is to avoid nested binding by changing, for example, the rule (*exp-med*):

$$(\widehat{y}P\widehat{\beta}\cdot\alpha)\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\gamma}\,[x]\ \widehat{z}R) \to Q\widehat{\gamma} \dagger \widehat{y}P\widehat{\beta} \dagger \widehat{z}R$$

Remember that the side-condition we need to be able to drop the parenthesis is $\beta \notin \mathit{fp}(Q)\backslash\{\gamma\}$, as well as that $y \notin \mathit{fs}(R)\backslash\{z\}$. But, to avoid an α-conflict, this is not enough. In fact, the α-conversion problem is generated in this rule (and here alone) by the fact that perhaps $\beta = \gamma$ or $y = z$, or β occurs bound in Q, or y in R. If so, these connectors

need to be renamed; one of the great plus points of $\mathcal{X}$ is that this can be done *within* the language itself, unlike for the λ-calculus. In fact, using Lemma 2.11, we can give an α-conflict free version of $\mathcal{X}$.

In contrast, notice that this is not possible for the λ-calculus. There the only reduction rule is $(\lambda x.M)N \to M[N/x]$, where the substitution is supposed to be *immediate*. In particular, it is impossible to predict (or even prevent) α-conflicts that typically depend on bindings occurring *inside* M and N.

To consider the substitution as a separate syntactic structure implies moving from the λ-calculus to $\lambda\mathbf{x}$. There the situation is slightly different, in that we can now say that

$$(\lambda y.M)\langle x=N\rangle \to \lambda z.(M\langle y=z\rangle\langle x=N\rangle),$$

thereby preventing a conflict on a possibly bound y in N. This is expensive though, as it is performed on *all* substitutions on abstractions, and does not actually detect the conflict, but just prevents it.

In $\mathcal{X}$, not only is the α-conflict solved, but also detected, all within the reduction system of $\mathcal{X}$ itself, by essentially expressing

$$(\lambda y.M)\langle x=N\rangle \to \lambda z.(M\langle y=z\rangle\langle x=N\rangle),\ \textit{if } y \textit{ bound in } N$$

As shown in [26], to accurately deal with α-conversion the rule (*exp-med*) needs to be replaced by:

$$\begin{aligned}
(\widehat{y}R\widehat{\beta}\cdot\alpha)\widehat{\alpha}\dagger\widehat{x}(Q\widehat{\gamma}\ [x]\ \widehat{z}P) &\to Q\widehat{\gamma}\dagger\widehat{y}(R\widehat{\beta}\dagger\widehat{z}P)\\
&\qquad \alpha, x \textit{ int.}, y\neq z, y\notin \textit{bs}(P)\\
(\widehat{y}R\widehat{\beta}\cdot\alpha)\widehat{\alpha}\dagger\widehat{x}(Q\widehat{\gamma}\ [x]\ \widehat{z}P) &\to Q\widehat{\gamma}\dagger\widehat{v}((\langle v.\delta\rangle\widehat{\delta}\nwarrow\widehat{y}R)\widehat{\beta}\dagger\widehat{z}P)\\
&\qquad \alpha, x \textit{ int.}, (y=z\vee y\in \textit{bs}(P)), v, \delta \textit{ fresh}\\
(\widehat{y}R\widehat{\beta}\cdot\alpha)\widehat{\alpha}\dagger\widehat{x}(Q\widehat{\gamma}\ [x]\ \widehat{z}P) &\to (Q\widehat{\gamma}\dagger\widehat{y}R)\widehat{\beta}\dagger\widehat{z}P\\
&\qquad \alpha, x \textit{ int.}, \gamma\neq\beta\notin \textit{bp}(Q)\\
(\widehat{y}R\widehat{\beta}\cdot\alpha)\widehat{\alpha}\dagger\widehat{x}(Q\widehat{\gamma}\ [x]\ \widehat{z}P) &\to (Q\widehat{\gamma}\dagger\widehat{y}(R\widehat{\beta}\nearrow\widehat{v}\langle v.\delta\rangle))\widehat{\delta}\dagger\widehat{z}P\\
&\qquad \alpha, x \textit{ int.}, (\beta=\gamma\vee\beta\in \textit{bp}(Q)), v, \delta \textit{ fresh}
\end{aligned}$$

It is even possible to check if y or β occur free in P, and to, if not, skip the renaming. Of course, other rules need dealing with as well. This modified reduction system preserves Barendregt's convention.

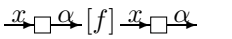
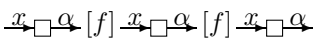
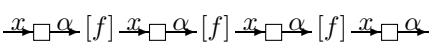
$0:$	$\langle x.\alpha\rangle$	
$1:$	$\langle x.\alpha\rangle\widehat{\alpha}\ [f]\ \widehat{x}\langle x.\alpha\rangle$	
$2:$	$\langle x.\alpha\rangle\widehat{\alpha}\ [f]\ \widehat{x}\langle x.\alpha\rangle\widehat{\alpha}\ [f]\ \widehat{x}\langle x.\alpha\rangle$	
$3:$	$\langle x.\alpha\rangle\widehat{\alpha}\ [f]\ \widehat{x}\langle x.\alpha\rangle\widehat{\alpha}\ [f]\ \widehat{x}\langle x.\alpha\rangle\widehat{\alpha}\ [f]\ \widehat{x}\langle x.\alpha\rangle$	

Figure 5: The first natural numbers in $\mathcal{X}$

3 The natural numbers into $\mathcal{X}$

The example of expressing natural numbers into $\mathcal{X}$ that we will give in this section is extremely interesting in two respects. Firstly, it shows how a basic structure can be embedded in $\mathcal{X}$. Secondly, it shows many features and among them alpha-conversion.

A natural number is represented in $\mathcal{X}$ by a sequence of capsules connected by mediating sockets, i.e., with the same used names. The first natural numbers are given in Figure 5.

We assume that natural numbers have two free sockets x and f and one free plug α, with x as an entering socket, α as an exiting plug and f as a mediating socket. We define 0 as $\langle x.\alpha\rangle$ and *succ*(N) as $N\widehat{\alpha}\,[f]\,\widehat{x}\langle x.\alpha\rangle$ where N itself is a natural number (which violates Barendregt's convention and should be removed in actual use). Clearly x and α have both a unique occurrence in each natural number. This is true for 0 and comes by induction for *succ*(N).

Lemma 3.1 *If N is a natural number,*

$$N\widehat{\alpha}\,[f]\,\widehat{x}\langle x.\alpha\rangle = \langle x.\alpha\rangle\widehat{\alpha}\,[f]\,\widehat{x}N.$$

Lemma 3.2 *If N_1 and N_2 are natural numbers,*

1. $(N_1\widehat{\alpha}\,[f]\,\widehat{x}\langle x.\alpha\rangle)\widehat{\alpha}\dagger\widehat{x}N_2 \;\to_{\mathrm{V}}\; N_1\widehat{\alpha}\,[f]\,\widehat{x}N_2$
2. $(N_1\widehat{\alpha}\,[f]\,\widehat{x}\langle x.\alpha\rangle)\widehat{\alpha}\dagger\widehat{x}N_2 \;\to_{\mathrm{N}}\; N_1\widehat{\alpha}\,[f]\,\widehat{x}N_2$
3. $N_1\widehat{\alpha}\dagger\widehat{x}(N_2\widehat{\alpha}\,[f]\,\widehat{x}\langle x.\alpha\rangle) \;\to_{\mathrm{V}}\; N_1\widehat{\alpha}\,[f]\,\widehat{x}N_2$
4. $N_1\widehat{\alpha}\dagger\widehat{x}(N_2\widehat{\alpha}\,[f]\,\widehat{x}\langle x.\alpha\rangle) \;\to_{\mathrm{N}}\; N_1\widehat{\alpha}\,[f]\,\widehat{x}N_2.$

Proof. In a first step, we rename bound variables to avoid confusion and capture. $=_\alpha$ is the alpha conversion. Let us prove *1*.

$$\begin{aligned}
&(N_1\widehat{\alpha}\,[f]\,\widehat{x}\langle x.\alpha\rangle)\widehat{\alpha}\dagger\widehat{x}N_2\\
&\quad=_\alpha\ (N_1[\beta/\alpha]\widehat{\beta}\,[f]\,\widehat{y}\langle y.\gamma\rangle)\widehat{\gamma}\dagger\widehat{x}N_2\\
&\quad\to_{\mathrm{V}}\ (N_1[\beta/\alpha]\widehat{\beta}\,[f]\,\widehat{y}\langle y.\gamma\rangle)\widehat{\gamma}\not\leftarrow\widehat{x}N_2\\
&\quad\to_{\mathrm{V}}\ (N_1[\beta/\alpha]\widehat{\gamma}\not\leftarrow\widehat{x}N_2)\widehat{\beta}\,[f]\,\widehat{y}(\langle y.\gamma\rangle\widehat{\gamma}\not\leftarrow\widehat{x}N_2)\\
&\quad\to_{\mathrm{V}}\ N_1[\beta/\alpha]\widehat{\beta}\,[f]\,\widehat{y}(\langle y.\gamma\rangle\widehat{\gamma}\not\leftarrow\widehat{x}N_2)\\
&\quad\to_{\mathrm{V}}\ N_1[\beta/\alpha]\widehat{\beta}\,[f]\,\widehat{y}N_2[x/y]\\
&\quad=_\alpha\ N_1\widehat{\alpha}\,[f]\,\widehat{x}N_2
\end{aligned}$$

Now we prove *4*.

$$\begin{aligned}
&N_1\widehat{\alpha}\dagger\widehat{x}(N_2\widehat{\alpha}\,[f]\,\widehat{x}\langle x.\alpha\rangle)\\
&\quad(3.1)\quad=\quad N_1\widehat{\alpha}\dagger\widehat{x}(\langle x.\alpha\rangle\widehat{\alpha}\,[f]\,\widehat{x}N_2)\\
&\qquad=_\alpha\ N_1[\beta/\alpha]\widehat{\beta}\dagger\widehat{y}(\langle y.\alpha\rangle\widehat{\alpha}\,[f]\,\widehat{x}N_2)\\
&\qquad\to_{\mathrm{N}}\ N_1[\beta/\alpha]\widehat{\beta}\nwarrow\widehat{y}(\langle y.\alpha\rangle\widehat{\alpha}\,[f]\,\widehat{x}N_2)\\
&\qquad\to_{\mathrm{N}}\ (N_1[\beta/\alpha]\widehat{\beta}\nwarrow\widehat{y}\langle y.\alpha\rangle)\widehat{\alpha}\,[f]\,\widehat{x}(N_1[\beta/\alpha]\widehat{\beta}\nwarrow\widehat{y}N_2)\\
&\qquad\to_{\mathrm{N}}\ N_1\widehat{\alpha}\,[f]\,\widehat{x}(N_1[\beta/\alpha]\widehat{\beta}\nwarrow\widehat{y}N_2)\\
&\qquad\to_{\mathrm{N}}\ N_1\widehat{\alpha}\,[f]\,\widehat{x}N_2
\end{aligned}$$

Part *2* and *3* come by induction using *1* and *4* and (*med-assoc*). □

We define now

Definition 3.3 (Addition and multiplication)

$$\begin{aligned}
\mathit{add}(N_1,N_2) &= N_1\widehat{\alpha}\dagger\widehat{x}N_2\\
\mathit{times}(N_1,N_2) &= (\widehat{x}N_1\widehat{\alpha}\cdot\beta)\widehat{\beta}\dagger\widehat{f}N_2
\end{aligned}$$

Using this definition, we can show that the normal properties for addition hold.

Lemma 3.4 (Properties of *add*)

$$\begin{aligned}
\mathit{add}(\langle x.\alpha\rangle, N) &\to_{\mathrm{A}} N\\
\mathit{add}(N, \langle x.\alpha\rangle) &\to_{\mathrm{A}} N\\
\mathit{add}(N_1\widehat{\alpha}\ [f]\ \widehat{x}\langle x.\alpha\rangle, N_2) &\to_{\mathrm{A}} \mathit{add}(N_1, N_2)\widehat{\alpha}\ [f]\ \widehat{x}\langle x.\alpha\rangle\\
\mathit{add}(N_1, N_2\widehat{\alpha}\ [f]\ \widehat{x}\langle x.\alpha\rangle) &\to_{\mathrm{A}} \mathit{add}(N_1, N_2)\widehat{\alpha}\ [f]\ \widehat{x}\langle x.\alpha\rangle
\end{aligned}$$

From Lemma 3.4 we get, by induction:

$$\begin{aligned}
\mathit{add}(0, N) &= N\\
\mathit{add}(N, 0) &= N\\
\mathit{add}(\mathit{succ}(N_1), N_2) &= \mathit{succ}(\mathit{add}(N_1, N_2)\\
\mathit{add}(\mathit{succ}(N_1), N_2) &= \mathit{succ}(\mathit{add}(N_1, N_2).
\end{aligned}$$

From them we can prove

$$\begin{aligned}
\mathit{add}(N_1, N_2) &= \mathit{add}(N_2, N_1)\\
\mathit{add}(N_1, \mathit{add}(N_2, N_3)) &= \mathit{add}(\mathit{add}(N_1, N_2), N_3).
\end{aligned}$$

Lemma 3.5 (Properties of *times*)

$$\begin{aligned}
\mathit{times}(N, 0) &\to_{\mathrm{A}} 0\\
\mathit{times}(N_1, \mathit{succ}(N_2)) &\to_{\mathrm{A}} \mathit{add}(\mathit{times}(N_1, N_2), N_1)
\end{aligned}$$

Proof. The first reduction is a consequences of Lemma 2.10. Let us prove the second reduction (in what follows N_1' stands for $N_1[y/x][\gamma/\alpha]$).

$$\begin{aligned}
&\mathit{times}(N_1, \mathit{succ}(N_2)) && =\\
&\mathit{times}(N_1, N_2\widehat{\alpha}\ [f]\ \widehat{x}\langle x.\alpha\rangle) && =\\
&(\widehat{x}N_1\widehat{\alpha}\cdot\beta)\widehat{\beta}\dagger\widehat{f}(N_2\widehat{\alpha}\ [f]\ \widehat{x}\langle x.\alpha\rangle) && =_{\alpha}\\
&(\widehat{y}N_1'\widehat{\gamma}\cdot\beta)\widehat{\beta}\dagger\widehat{f}(N_2\widehat{\alpha}\ [f]\ \widehat{x}\langle x.\alpha\rangle) && \to_{\mathrm{N}}\\
&(\widehat{y}N_1'\widehat{\gamma}\cdot\beta)\widehat{\beta}\nwarrow\widehat{f}(N_2\widehat{\alpha}\ [f]\ \widehat{x}\langle x.\alpha\rangle) && \to_{\mathrm{N}}\\
&(\widehat{y}N_1'\widehat{\gamma}\cdot\beta)\widehat{\beta}\dagger\widehat{g}(((\widehat{y}N_1'\widehat{\gamma}\cdot\beta)\widehat{\beta}\nwarrow\widehat{f}N_2)\widehat{\alpha}\ [g]\ \widehat{x}((\widehat{y}N_1'\widehat{\gamma}\cdot\beta)\widehat{\beta}\nwarrow\widehat{f}\langle x.\alpha\rangle)) && \to_{\mathrm{N}}\\
&(\widehat{y}N_1'\widehat{\gamma}\cdot\beta)\widehat{\beta}\dagger\widehat{g}(((\widehat{y}N_1'\widehat{\gamma}\cdot\beta)\widehat{\beta}\nwarrow\widehat{f}N_2)\widehat{\alpha}\ [g]\ \widehat{x}\langle x.\alpha\rangle) && \to_{\mathrm{N}}\\
&((\widehat{y}N_1'\widehat{\gamma}\cdot\beta)\widehat{\beta}\nwarrow\widehat{f}N_2)\widehat{\alpha}\dagger\widehat{y}N_1'\widehat{\gamma}\dagger\widehat{x}\langle x.\alpha\rangle && \to_{\mathrm{A}}\\
&((\widehat{y}N_1'\widehat{\gamma}\cdot\beta)\widehat{\beta}\nwarrow\widehat{f}N_2)\widehat{\alpha}\dagger\widehat{y}N_1'[\alpha/\gamma] && =_{\alpha}\\
&((\widehat{x}N_1\widehat{\alpha}\cdot\beta)\widehat{\beta}\nwarrow\widehat{f}N_2)\widehat{\alpha}\dagger\widehat{x}N_1 && =\\
&(\mathit{times}(N_1, N_2))\widehat{\alpha}\dagger\widehat{x}N_1 && =\\
&\mathit{add}(\mathit{times}(N_1, N_2), N_1) &&
\end{aligned}$$

□

4 Interpreting $\lambda\mathbf{x}$

After natural numbers, a good candidate is the λ-calculus. Actually, instead of λ-calculus, we directly interpret a calculus of explicit substitutions, namely $\lambda\mathbf{x}$.

$\lambda\mathbf{x}$ is a calculus of explicit substitutions introduced by Bloo and Rose [7], where any β-reduction of λ-calculus can be split into several more atomic steps of computation. In this section we show that $\mathcal{X}$ has a lower level of atomicity as it simulates each reduction step by describing how the explicit substitutions interact with terms.

We briefly recall here the calculus $\lambda\mathbf{x}$. Its syntax is an extension of that of λ-calculus:

$$M ::= x \mid \lambda x.M \mid M_1 M_2 \mid M\langle x = N\rangle$$

Definition 4.1 ($\lambda\mathbf{x}$) *The reduction relation is defined by the following rules*

$$\begin{array}{rcll}
(\lambda x.M)P & \to & M\langle x=P\rangle & (\mathsf{B})\\
(MN)\langle x=P\rangle & \to & M\langle x=P\rangle N\langle x=P\rangle & (\mathsf{App})\\
(\lambda y.M)\langle x=P\rangle & \to & \lambda y.(M\langle x=P\rangle) & (\mathsf{Abs})\\
x\langle x=P\rangle & \to & P & (\mathsf{VarI})\\
y\langle x=P\rangle & \to & y & (\mathsf{VarK})\\
M\langle x=P\rangle & \to & M, \text{ if } x\notin \mathit{fv}(M) & (\mathsf{gc})
\end{array}$$

The notion of reduction $\lambda\mathbf{x}$ is obtained by deleting rule (gc), and the notion of reduction $\lambda\mathbf{x}_{\mathsf{gc}}$ is obtained by deleting rule (VarK). The rule (gc) is called 'garbage collection', as it removes useless substitutions.

Definition 4.2 (Interpretation of $\lambda\mathbf{x}$ in $\mathcal{X}$)

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

The interpretation of the λ-calculus is just made of the first three rules.

Now we show that the reductions can be simulated, preserving the evaluation strategies. But what is the CBV-$\lambda\mathbf{x}$? Our notion is naturally inspired by that of the λ-calculus: in a CBV-β-reduction, the argument must be a value, so that means that when it is simulated by the CBV-$\lambda\mathbf{x}$, all the substitutions created are of the form $M\langle x=N\rangle$ where N is a value, that is, either a variable or an abstraction, just as in λ-calculus. Hence, we build the CBV-$\lambda\mathbf{x}$ by a syntactic restriction:

$$M ::= x \mid \lambda x.M \mid M_1 M_2 \mid M\langle x=\lambda x.N\rangle \mid M\langle x=y\rangle$$

Now notice that, again, N is a value if and only if $[\![N]\!]^{\mathrm{x}}_{\alpha}$ introduces α.

Theorem 4.3 CBN**:** $[\![(\lambda x.M)N]\!]^{\mathrm{x}}_{\alpha} \to_{\mathrm{N}} [\![M\langle x=N\rangle]\!]^{\mathrm{x}}_{\alpha}$

CBV**:** $[\![(\lambda x.M)N]\!]^{\mathrm{x}}_{\alpha} \to_{\mathrm{V}} [\![M\langle x=N\rangle]\!]^{\mathrm{x}}_{\alpha}$ *iff N is a value (that is, if and only if $(\lambda x.M)N \to_{\mathrm{V}} M\langle x=N\rangle$).*

Proof.
$$\begin{array}{lll}
[\![(\lambda x.M)N]\!]^{\mathrm{x}}_{\alpha} & \triangleq & \\
[\![\lambda x.M]\!]^{\mathrm{x}}_{\gamma}\widehat{\gamma}\dagger\widehat{y}([\![N]\!]^{\mathrm{x}}_{\beta}\widehat{\beta}\ [y]\ \widehat{z}\langle z.\alpha\rangle) & \triangleq & \\
(\widehat{x}[\![M]\!]^{\mathrm{x}}_{\delta}\widehat{\delta}\cdot\gamma)\widehat{\gamma}\dagger\widehat{y}([\![N]\!]^{\mathrm{x}}_{\beta}\widehat{\beta}\ [y]\ \widehat{z}\langle z.\alpha\rangle) & \to & (\textit{exp-med})\\
[\![N]\!]^{\mathrm{x}}_{\beta}\widehat{\beta}\dagger\widehat{x}([\![M]\!]^{\mathrm{x}}_{\delta}\widehat{\delta}\dagger\widehat{z}\langle z.\alpha\rangle) & \to & (\textit{bend-}\mathrm{R})\\
[\![N]\!]^{\mathrm{x}}_{\beta}\widehat{\beta}\backslash\widehat{x}([\![M]\!]^{\mathrm{x}}_{\delta}\widehat{\delta}\dagger\widehat{z}\langle z.\alpha\rangle) & \to & (\textit{2.11-2})\\
[\![N]\!]^{\mathrm{x}}_{\beta}\widehat{\beta}\backslash\widehat{x}[\![M]\!]^{\mathrm{x}}_{\alpha} & &
\end{array}$$

Notice that this reduction sequence is valid in the CBN-evaluation, which proves point 1. As for the CBV-evaluation, the step $(\textit{bend}\text{-R})$ is only possible if $\llbracket N \rrbracket^{\mathrm{x}}_{\beta}$ introduces β, that is, if N is a value. If N is not a value, then $\llbracket N \rrbracket^{\mathrm{x}}_{\beta}$ does not introduce β. The active weld $\llbracket N \rrbracket^{\mathrm{x}}_{\beta} \widehat{\beta} \nwarrow \widehat{x} \llbracket M \rrbracket^{\mathrm{x}}_{\alpha}$ that has to be reached was either already in $\llbracket (\lambda x.M) N \rrbracket^{\mathrm{x}}_{\alpha}$ or was created (by rule $(\textit{bend}\text{-R})$) along the reduction. The first case is impossible, since the weld should either be in $\llbracket N \rrbracket^{\mathrm{x}}_{\beta}$ or in $\llbracket M \rrbracket^{\mathrm{x}}_{\delta}$ which are both smaller in size than the weld. The second case is also impossible because in the CBV-evaluation, rule $(\textit{bend}\text{-R})$ requires $\llbracket N \rrbracket^{\mathrm{x}}_{\beta}$ to introduce β. □

Theorem 4.4 (Simulation of the other rules)
If $M \to_{\mathrm{x}} N$ *then* $\llbracket M \rrbracket^{\mathrm{x}}_{\gamma} \to_{\mathrm{v}} \llbracket N \rrbracket^{\mathrm{x}}_{\gamma}$ *and* $\llbracket M \rrbracket^{\mathrm{x}}_{\gamma} \to_{\mathrm{N}} \llbracket N \rrbracket^{\mathrm{x}}_{\gamma}$

Proof.

- $\llbracket (PQ)\langle x = N \rangle \rrbracket^{\mathrm{x}}_{\alpha} \to \llbracket (P\langle x = N \rangle)(Q\langle x = N \rangle) \rrbracket^{\mathrm{x}}_{\alpha}$

$$
\begin{array}{ll}
\llbracket (PQ)\langle x = N \rangle \rrbracket^{\mathrm{x}}_{\alpha} \triangleq \llbracket N \rrbracket^{\mathrm{x}}_{\beta} \widehat{\beta} \nwarrow \widehat{x} \llbracket PQ \rrbracket^{\mathrm{x}}_{\alpha} & \triangleq \\
\llbracket N \rrbracket^{\mathrm{x}}_{\beta} \widehat{\beta} \nwarrow \widehat{x} (\llbracket P \rrbracket^{\mathrm{x}}_{\gamma} \widehat{\gamma} \dagger \widehat{y} (\llbracket Q \rrbracket^{\mathrm{x}}_{\beta} \widehat{\beta}\ [y]\ \widehat{z}\langle z.\alpha \rangle)) & \to (\nwarrow \textit{weld}) \\
(\llbracket N \rrbracket^{\mathrm{x}}_{\beta} \widehat{\beta} \nwarrow \widehat{x} \llbracket P \rrbracket^{\mathrm{x}}_{\gamma}) \widehat{\gamma} \dagger \widehat{y} (\llbracket N \rrbracket^{\mathrm{x}}_{\beta} \widehat{\beta} \nwarrow \widehat{x} (\llbracket Q \rrbracket^{\mathrm{x}}_{\beta} \widehat{\beta}\ [y]\ \widehat{z}\langle z.\alpha \rangle))) & \\
& \to (\nwarrow \textit{med2}) \\
\llbracket P\langle x = N \rangle \rrbracket^{\mathrm{x}}_{\gamma} \widehat{\gamma} \dagger \widehat{y} ((\llbracket N \rrbracket^{\mathrm{x}}_{\beta} \widehat{\beta} \nwarrow \widehat{x} \llbracket Q \rrbracket^{\mathrm{x}}_{\beta}) \widehat{\beta}\ [y]\ \widehat{z} (\llbracket N \rrbracket^{\mathrm{x}}_{\beta} \widehat{\beta} \nwarrow \widehat{x}\langle z.\alpha \rangle))) & \\
& \triangleq \\
\llbracket P\langle x = N \rangle \rrbracket^{\mathrm{x}}_{\gamma} \widehat{\gamma} \dagger \widehat{y} (\llbracket Q\langle N = x \rangle \rrbracket^{\mathrm{x}}_{\beta} \widehat{\beta}\ [y]\ \widehat{z} (\llbracket N \rrbracket^{\mathrm{x}}_{\beta} \widehat{\beta} \nwarrow \widehat{x}\langle z.\alpha \rangle))) & \\
& \to (\not\nwarrow \textit{cap}) \\
\llbracket P\langle x = N \rangle \rrbracket^{\mathrm{x}}_{\gamma} \widehat{\gamma} \dagger \widehat{y} (\llbracket Q\langle N = x \rangle \rrbracket^{\mathrm{x}}_{\beta} \widehat{\beta}\ [y]\ \widehat{z}\langle z.\alpha \rangle) & \triangleq \\
\llbracket (P\langle x = N \rangle)(Q\langle x = N \rangle) \rrbracket^{\mathrm{x}}_{\alpha} &
\end{array}
$$

- $\llbracket (\lambda y.M)\langle x = N \rangle \rrbracket^{\mathrm{x}}_{\alpha} \to \llbracket \lambda y.M\langle x = N \rangle \rrbracket^{\mathrm{x}}_{\alpha}$

$$
\begin{array}{lll}
\llbracket (\lambda y.M)\langle x = N \rangle \rrbracket^{\mathrm{x}}_{\alpha} & \triangleq \llbracket N \rrbracket^{\mathrm{x}}_{\beta} \widehat{\beta} \nwarrow \widehat{x} \llbracket \lambda y.M \rrbracket^{\mathrm{x}}_{\alpha} & \triangleq \\
\llbracket N \rrbracket^{\mathrm{x}}_{\beta} \widehat{\beta} \nwarrow \widehat{x} (\widehat{y} \llbracket M \rrbracket^{\mathrm{x}}_{\gamma} \widehat{\gamma} \cdot \alpha) & \to (\nwarrow \textit{exp}) & \\
\widehat{y} (\llbracket N \rrbracket^{\mathrm{x}}_{\beta} \widehat{\beta} \nwarrow \widehat{x} \llbracket M \rrbracket^{\mathrm{x}}_{\gamma}) \widehat{\gamma} \cdot \alpha & \triangleq & \\
\widehat{y} \llbracket M\langle x = N \rangle \rrbracket^{\mathrm{x}}_{\gamma} \widehat{\gamma} \cdot \alpha & \triangleq \llbracket \lambda y.M\langle x = N \rangle \rrbracket^{\mathrm{x}}_{\alpha}. &
\end{array}
$$

- $\llbracket x\langle x = N \rangle \rrbracket^{\mathrm{x}}_{\alpha} \to \llbracket N \rrbracket^{\mathrm{x}}_{\alpha}$.

$$
\begin{array}{lll}
\llbracket x\langle x = N \rangle \rrbracket^{\mathrm{x}}_{\alpha} & \triangleq \llbracket N \rrbracket^{\mathrm{x}}_{\beta} \widehat{\beta} \nwarrow \widehat{x} \llbracket x \rrbracket^{\mathrm{x}}_{\alpha} & \triangleq \\
\llbracket N \rrbracket^{\mathrm{x}}_{\beta} \widehat{\beta} \nwarrow \widehat{x}\langle x.\alpha \rangle & \to (\textit{d}\text{R}) & \\
\llbracket N \rrbracket^{\mathrm{x}}_{\beta} \widehat{\beta} \dagger \widehat{x}\langle x.\alpha \rangle & \to (\textit{2.11-2})\ \llbracket N \rrbracket^{\mathrm{x}}_{\alpha} &
\end{array}
$$

- $\llbracket y\langle x = N \rangle \rrbracket^{\mathrm{x}}_{\alpha} \to \llbracket y \rrbracket^{\mathrm{x}}_{\alpha}$.

$$
\begin{array}{lll}
\llbracket y\langle x = N \rangle \rrbracket^{\mathrm{x}}_{\alpha} & \triangleq \llbracket N \rrbracket^{\mathrm{x}}_{\beta} \widehat{\beta} \nwarrow \widehat{x} \llbracket y \rrbracket^{\mathrm{x}}_{\alpha} & \triangleq \\
\llbracket N \rrbracket^{\mathrm{x}}_{\beta} \widehat{\beta} \nwarrow \widehat{x}\langle y.\alpha \rangle & \to (\not\nwarrow \textit{cap})\ \langle y.\alpha \rangle & \triangleq \llbracket y \rrbracket^{\mathrm{x}}_{\alpha}
\end{array}
$$

- $\llbracket M\,\langle x=N\rangle \rrbracket^{\mathrm{x}}_{\alpha} \to \llbracket M \rrbracket^{\mathrm{x}}_{\alpha}$, if $x \notin \mathit{fv}(M)$.

$$\llbracket M\,\langle x=N\rangle \rrbracket^{\mathrm{x}}_{\alpha} \triangleq \llbracket N \rrbracket^{\mathrm{x}}_{\beta} \widehat{\beta}\backslash\widehat{x} \llbracket M \rrbracket^{\mathrm{x}}_{\alpha} \to (\mathit{gc}\text{-R})\ \llbracket M \rrbracket^{\mathrm{x}}_{\alpha}$$

□

Hence, we deduce the fact that reduction in $\lambda\mathbf{x}$ is preserved by the interpretation of terms into $\mathcal{X}$.

Theorem 4.5 (Simulation of λx)

1. *If* $M \to_{\mathrm{v}} N$ *then* $\llbracket M \rrbracket^{\mathrm{x}}_{\gamma} \to_{\mathrm{v}}{}^{*} \llbracket N \rrbracket^{\mathrm{x}}_{\gamma}$
2. *If* $M \to_{\mathrm{N}} N$ *then* $\llbracket M \rrbracket^{\mathrm{x}}_{\gamma} \to_{\mathrm{N}}{}^{*} \llbracket N \rrbracket^{\mathrm{x}}_{\gamma}$

5 Typing for $\mathcal{X}$

The notion of type assignment on $\mathcal{X}$ that we present in this section is the basic implicative system for Classical Logic (Gentzen system LK)g. The Curry-Howard property is easily achieved (see Section 10).

Definition 5.1 (Types and Contexts)

1. *The set of types is defined by the grammar:*

$$A, B ::= \varphi \mid A{\to}B$$

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

2. *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. 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 $A_1 = A_2$).*

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

Definition 5.2 (Typing for $\mathcal{X}$) *1.* Type judgements *are expressed via a ternary relation $P \vdots \Gamma \vdash \Delta$, 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.*

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

$$(\mathit{cap}) : \frac{}{\langle y.\alpha\rangle \vdots \Gamma, y{:}A \vdash \alpha{:}A, \Delta}$$

$$(\mathit{exp}) : \frac{P \vdots \Gamma, x{:}A \vdash \alpha{:}B, \Delta}{\widehat{x}P\widehat{\alpha}{\cdot}\beta \vdots \Gamma \vdash \beta{:}A{\to}B, \Delta}$$

$$(\mathit{med}) : \frac{P \vdots \Gamma \vdash \alpha{:}A, \Delta \quad Q \vdots \Gamma, x{:}B \vdash \Delta}{P\widehat{\alpha}\,[y]\,\widehat{x}Q \vdots \Gamma, y{:}A{\to}B \vdash \Delta}$$

$$(\mathit{cut}) : \frac{P \vdots \Gamma \vdash \alpha{:}A, \Delta \quad Q \vdots \Gamma, x{:}A \vdash \Delta}{P\widehat{\alpha} \dagger \widehat{x}Q \vdots \Gamma \vdash \Delta}$$

We write $P \vdots \Gamma \vdash \Delta$ if there exists a derivation that has this judgement in the bottom line, and write $D :: P \vdots \Gamma \vdash \Delta$ if we want to name that derivation.

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.

We can now provide the type of two kinds of term we already studied.

- The type of natural numbers in $\mathcal{X}$ is $N \vdots x{:}A, f{:}A \to A \vdash \alpha{:}A$.
- The type of $[\![M]\!]^{\mathrm{x}}_{\alpha}$ is $M \vdots \Gamma \vdash \alpha{:}A$ if $\Gamma \vdash_{\lambda} M : A$.

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

Theorem 5.3 (Subject reduction) *If $P \vdots \Gamma \vdash \Delta$, and $P \to Q$, then $Q \vdots \Gamma \vdash \Delta$.*

Theorem 5.4 (Strong normalisation [25])
If $P \vdots \Gamma \vdash \Delta$, then P is strongly normalising.

Theorem 5.5 (Subject congruence) *If $P \vdots \Gamma \vdash \Delta$, and $P \overset{\mathcal{X}}{=} Q$, then $Q \vdots \Gamma \vdash \Delta$.*

Proof. We will only prove the result for the rule (*med-assoc*). Consider the two following typing trees:

- for the term $(P_1\widehat{\alpha}\dagger\widehat{x}P_2)\widehat{\beta}\dagger\widehat{y}P_3$

$$\dfrac{\dfrac{P_1 \vdots \Gamma \vdash \beta{:}B, \alpha : A, \Delta \quad P_2 \vdots \Gamma, x{:}A \vdash \beta{:}B, \Delta}{P_1\widehat{\alpha}\dagger\widehat{x}P_2 \vdots \Gamma \vdash \beta{:}B, \Delta} \qquad P_3 \vdots \Gamma, y : B \vdash \Delta}{(P_1\widehat{\alpha}\dagger\widehat{x}P_2)\widehat{\beta}\dagger\widehat{y}P_3 \vdots \Gamma \vdash \Delta}$$

- and for the term $P_1\widehat{\alpha}\dagger\widehat{x}(P_2\widehat{\beta}\dagger\widehat{y}P_3)$

$$\dfrac{P_1 \vdots \Gamma \vdash \alpha{:}A, \Delta \qquad \dfrac{P_2 \vdots \Gamma, x{:}A \vdash \Delta, \beta{:}B \quad P_2 \vdots \Gamma, x{:}A, y{:}B \vdash \Delta}{P_2\widehat{\beta}\dagger\widehat{y}P_3 \vdots \Gamma, x{:}A \vdash \Delta}}{P_1\widehat{\alpha}\dagger\widehat{x}(P_2\widehat{\beta}\dagger\widehat{y}P_3) \vdots \Gamma \vdash \Delta}$$

But the preconditions $\beta \notin \mathit{fp}(P_1)$ for *med-assoc* allows us to conclude $P_1 \vdots \Gamma \vdash \Delta, \beta{:}B, \alpha{:}A$ from $P_1 \vdots \Gamma \vdash \Delta, \alpha{:}A$, by weakening and the same for P_3 hence the *Subject congruence.* □

6 Interpreting the $\lambda\mu\tilde{\mu}$-calculus

In its typed version, $\mathcal{X}$ is a proof-term syntax for a classical sequent calculus. Another proof-system has been proposed for (a variant of) classical sequent calculus: Curien and Herbelin's $\lambda\mu\tilde{\mu}$-calculus [10]. It is interesting to relate those two formalisms and realize that $\lambda\mu\tilde{\mu}$ can be interpreted in $\mathcal{X}$ as well.

Definition 6.1 (Syntax and Reduction $\lambda\mu\tilde{\mu}$)

$$\begin{array}{lll} c ::= \langle v|e\rangle & & (\textit{commands}) \\ v ::= x \mid \mu\beta.c \mid \lambda x.v & & (\textit{terms}) \\ e ::= \alpha \mid \tilde{\mu}x.c \mid v \cdot e & & (\textit{contexts}) \end{array}$$

$$
\begin{array}{lll}
(\to) & \langle \lambda x.v_1 | v_2 \cdot e \rangle & \to \langle v_2 | \tilde{\mu} x.\langle v_1 | e \rangle \rangle \\
(\mu) & \langle \mu\beta.c | e \rangle & \to c[e/\beta] \\
(\tilde{\mu}) & \langle v | \tilde{\mu} x.e \rangle & \to c[v/x]
\end{array}
$$

Definition 6.2 (Typing for $\lambda\mu\tilde{\mu}$)

$$
(\mathit{cut}) : \frac{\Gamma \vdash v{:}A \mid \Delta \quad \Gamma \mid e{:}A \vdash \Delta}{\langle v|e\rangle : \Gamma \vdash \Delta}
$$

$$
(\mathit{Ax}\text{-}n) : \frac{}{\Gamma \mid x{:}A \vdash x{:}A, \Delta} \quad (\mathit{RI}) : \frac{\Gamma, x{:}A \vdash v{:}B \mid \Delta}{\Gamma \vdash \lambda x.v{:}A{\to}B \mid \Delta}
$$

$$
(\mathit{Ax}\text{-}v) : \frac{}{\Gamma, \alpha{:}A \vdash \alpha{:}A \mid \Delta} \quad (\mathit{LI}) : \frac{\Gamma \vdash v{:}A \mid \Delta \quad \Gamma \mid e{:}B \vdash \Delta}{\Gamma \mid v \cdot e{:}A{\to}B \vdash \Delta}
$$

$$
(\mu) : \frac{c : \Gamma \vdash \alpha{:}A, \Delta}{\Gamma \vdash \mu\alpha.c{:}A \mid \Delta} \qquad (\tilde{\mu}) : \frac{c : \Gamma, \alpha{:}A \vdash \Delta}{\Gamma \mid \tilde{\mu}\alpha.c{:}A \vdash \Delta}
$$

One feature of $\lambda\mu\tilde{\mu}$ is that its syntax and reduction rules reveal a duality between call-by-value and call-by-name. Indeed, the call-by-value evaluation is obtained by forbidding a $\tilde{\mu}$-reduction when the redex is also a μ-redex, whereas the call-by-name evaluation forbids a μ-reduction when the redex is also a $\tilde{\mu}$-redex.

We show here how $\mathcal{X}$ accounts for this duality. A simulation of $\lambda\mu\tilde{\mu}$ by $\mathcal{X}$ has already been proposed in [19], but we strengthen the result by stating that this simulation preserves the CBV and CBN evaluations:

Definition 6.3 (Translation of $\lambda\mu\tilde{\mu}$ into $\mathcal{X}$)

$$
\begin{array}{rcl}
\llbracket x \rrbracket_{\alpha}^{\lambda\mu\tilde{\mu}} & = & \langle x.\alpha \rangle \\
\llbracket \lambda x.v \rrbracket_{\alpha}^{\lambda\mu\tilde{\mu}} & = & \widehat{x} \llbracket v \rrbracket_{\beta}^{\lambda\mu\tilde{\mu}} \widehat{\beta} \cdot \alpha \\
\llbracket \mu\beta.c \rrbracket_{\alpha}^{\lambda\mu\tilde{\mu}} & = & \llbracket c \rrbracket^{\lambda\mu\tilde{\mu}} \widehat{\beta} \dagger \widehat{x} \langle x.\alpha \rangle
\end{array}
$$

$$
\begin{array}{rcl}
\llbracket \alpha \rrbracket_{x}^{\lambda\mu\tilde{\mu}} & = & \langle x.\alpha \rangle \\
\llbracket v \cdot e \rrbracket_{x}^{\lambda\mu\tilde{\mu}} & = & \llbracket v \rrbracket_{\alpha}^{\lambda\mu\tilde{\mu}} \widehat{\alpha}\ [x]\ \widehat{y} \llbracket e \rrbracket_{y}^{\lambda\mu\tilde{\mu}} \\
\llbracket \tilde{\mu} y.c \rrbracket_{x}^{\lambda\mu\tilde{\mu}} & = & \langle x.\beta \rangle \widehat{\beta} \dagger \widehat{y} \llbracket c \rrbracket^{\lambda\mu\tilde{\mu}}
\end{array}
$$

$$
\llbracket \langle v|e \rangle \rrbracket^{\lambda\mu\tilde{\mu}} = \llbracket v \rrbracket_{\alpha}^{\lambda\mu\tilde{\mu}} \widehat{\alpha} \dagger \widehat{x} \llbracket e \rrbracket_{x}^{\lambda\mu\tilde{\mu}}
$$

The interpretation function preserves typeability:

Lemma 6.4 ([19])

1. *If* $\Gamma \vdash v{:}A \mid \Delta$, *then* $\llbracket v \rrbracket_{\alpha}^{\lambda\mu\tilde{\mu}} : \Gamma \vdash \alpha{:}A, \Delta$.
2. *If* $\Gamma \mid e{:}A \vdash \Delta$, *then* $\llbracket e \rrbracket_{x}^{\lambda\mu\tilde{\mu}} : \Gamma, x{:}A \vdash \Delta$.
3. *If* $c : \Gamma \vdash \Delta$, *then* $\llbracket c \rrbracket^{\lambda\mu\tilde{\mu}} : \Gamma \vdash \Delta$.

Theorem 6.5 (Simulation of $\lambda\mu\tilde{\mu}$)

1. *If* $c \to_{\mathrm{V}} c'$ *then* $\llbracket c \rrbracket^{\lambda\mu\tilde{\mu}} \to_{\mathrm{V}} \llbracket c' \rrbracket^{\lambda\mu\tilde{\mu}}$
2. *If* $c \to_{\mathrm{N}} c'$ *then* $\llbracket c \rrbracket^{\lambda\mu\tilde{\mu}} \to_{\mathrm{N}} \llbracket c' \rrbracket^{\lambda\mu\tilde{\mu}}$

7 Interpreting $\lambda\mu$

Parigot's $\lambda\mu$-calculus [24] is yet another proof-term syntax for classical logic, but expressed in the setting of Natural Deduction. Curien and Herbelin have shown how the normalisation in $\lambda\mu$ can be interpreted as the cut-elimination in $\lambda\mu\tilde{\mu}$. But to reflect the CBV-evaluation and CBN-evaluation, they define two encodings, respectively:

Definition 7.1 ([10]) *The* CBV*-interpretation* $\cdot^{<}$ *and* CBN*-interpretation* $\cdot^{>}$ *of* $\lambda\mu$ *are defined as follows:*

$$\begin{aligned}
x^{<} &= x \\
(\lambda x.M)^{<} &= \lambda x.M^{<} \\
(MN)^{<} &= \mu\alpha.\langle N^{<}|\tilde{\mu}x.\langle M^{>}|x\cdot\alpha\rangle\rangle \\
(\mu\beta.c)^{<} &= \mu\beta.c^{<} \\
([\alpha]M)^{<} &= \langle M^{<}|\alpha\rangle
\end{aligned}$$

$$\begin{aligned}
x^{>} &= x \\
(\lambda x.M)^{>} &= \lambda x.M^{>} \\
(MN)^{>} &= \mu\alpha.\langle M^{>}|N^{>}\cdot\alpha\rangle \\
(\mu\beta.c)^{>} &= \mu\beta.c^{>} \\
([\alpha]M)^{>} &= \langle M^{>}|\alpha\rangle
\end{aligned}$$

Curien and Herbelin's result is:

Theorem 7.2 (Simulation of $\lambda\mu$ in $\lambda\mu\tilde{\mu}$ [10])

1. *If* $t \to_{\mathrm{V}} t'$ *then* $t^{<} \to_{\mathrm{V}}{}^{*} t'^{<}$.
2. *If* $t \to_{\mathrm{N}} t'$ *then* $t^{>} \to_{\mathrm{N}}{}^{*} t'^{>}$.

Hence, combining this result to Theorem 6.5 we also get:

Theorem 7.3 (Simulation of $\lambda\mu$ in $\mathcal{X}$)

1. *If* $t \to_{\mathrm{V}} t'$ *then* $[\![t^{<}]\!]^{\lambda\mu\tilde{\mu}} \to_{\mathrm{V}}{}^{*} [\![t'^{<}]\!]^{\lambda\mu\tilde{\mu}}$.
2. *If* $t \to_{\mathrm{N}} t'$ *then* $[\![t^{>}]\!]^{\lambda\mu\tilde{\mu}} \to_{\mathrm{N}}{}^{*} [\![t'^{>}]\!]^{\lambda\mu\tilde{\mu}}$.

8 Improving the interpretation of the λ-calculus

Theorem 7.2 also holds for the restriction of $\lambda\mu$ to the traditional λ-calculus. However, one might be disappointed that the preservation of the CBV-evaluation and CBN-evaluation relies on two *distinct* translations of terms. For instance, the CBV-λ-calculus and the CBN-λ-calculus can both be encoded into CPS [3], and there it is clear that what accounts for the distinction CBV/CBN is the encodings themselves, and not the way CPS reduces the encoded terms.

So it could be the case that, similarly, when encoding the λ-calculus in $\lambda\mu\tilde{\mu}$, the distinction between CBV and CBN mostly relies on Curien and Herbelin's two distinct encodings rather than the features of $\lambda\mu\tilde{\mu}$. The same holds for [27]. Whereas their CBN-translation seems intuitive, they apparently need to twist it in a more complex way in order to give an accurate interpretation of the CBV λ-calculus, since

Lemma 8.1 $M^{<} \to^{*} M^{>}$

$$
\begin{array}{ll}
[\![\Delta\Delta]\!]^{\mathrm{N}}_{\beta} \triangleq [\![\lambda x.xx]\!]^{\mathrm{N}}_{\gamma}\widehat{\gamma}\dagger\widehat{z}([\![\lambda x.xx]\!]^{\mathrm{N}}_{\gamma}\widehat{\gamma}\ [z]\ \widehat{y}\langle y.\beta\rangle) & \triangleq \\
(\widehat{x}[\![xx]\!]^{\mathrm{N}}_{\alpha}\widehat{\alpha}\cdot\delta)\widehat{\delta}\dagger\widehat{z}([\![\lambda x.xx]\!]^{\mathrm{N}}_{\gamma}\widehat{\gamma}\ [z]\ \widehat{y}\langle y.\beta\rangle) & \to (\textit{exp-med}) \\
[\![\lambda x.xx]\!]^{\mathrm{N}}_{\gamma}\widehat{\gamma}\dagger\widehat{x}([\![xx]\!]^{\mathrm{N}}_{\alpha}\widehat{\alpha}\dagger\widehat{y}\langle y.\beta\rangle) & \to (\textit{2.11-2}) \\
[\![\lambda x.xx]\!]^{\mathrm{N}}_{\gamma}\widehat{\gamma}\dagger\widehat{x}[\![xx]\!]^{\mathrm{N}}_{\beta} & \triangleq \\
[\![\lambda x.xx]\!]^{\mathrm{N}}_{\gamma}\widehat{\gamma}\dagger\widehat{x}(\langle x.\delta\rangle\widehat{\delta}\ [x]\ \widehat{y}\langle y.\beta\rangle) & \to (\textit{bend-R}) \\
[\![\lambda x.xx]\!]^{\mathrm{N}}_{\gamma}\widehat{\gamma}\nwarrow\widehat{x}(\langle x.\delta\rangle\widehat{\delta}\ [x]\ \widehat{y}\langle y.\beta\rangle) & \to (\nwarrow \textit{med1}) \\
[\![\lambda x.xx]\!]^{\mathrm{N}}_{\gamma}\widehat{\gamma}\dagger\widehat{z}((\![\lambda x.xx]\!]^{\mathrm{N}}_{\gamma}\widehat{\gamma}\nwarrow\widehat{x}\langle x.\delta\rangle)\widehat{\delta}\ [z]\ \widehat{y}([\![\lambda x.xx]\!]^{\mathrm{N}}_{\gamma}\widehat{\gamma}\nwarrow\widehat{x}\langle y.\beta\rangle)) & \to (\nearrow \textit{cap}) \\
[\![\lambda x.xx]\!]^{\mathrm{N}}_{\gamma}\widehat{\gamma}\dagger\widehat{z}((\![\lambda x.xx]\!]^{\mathrm{N}}_{\gamma}\widehat{\gamma}\nwarrow\widehat{x}\langle x.\delta\rangle)\widehat{\delta}\ [z]\ \widehat{y}\langle y.\beta\rangle) & \to (\nwarrow\dagger) \\
[\![\lambda x.xx]\!]^{\mathrm{N}}_{\gamma}\widehat{\gamma}\dagger\widehat{z}((\![\lambda x.xx]\!]^{\mathrm{N}}_{\gamma}\widehat{\gamma}\dagger\widehat{x}\langle x.\delta\rangle)\widehat{\delta}\ [z]\ \widehat{y}\langle y.\beta\rangle) & \to (\textit{2.11-2}) \\
[\![\lambda x.xx]\!]^{\mathrm{N}}_{\gamma}\widehat{\gamma}\dagger\widehat{z}([\![\lambda x.xx]\!]^{\mathrm{N}}_{\delta}\widehat{\delta}\ [z]\ \widehat{y}\langle y.\alpha\rangle) & \triangleq [\![\Delta\Delta]\!]^{\mathrm{N}}_{\beta}
\end{array}
$$

Figure 6: Reduction of the interpretation of the lambda term $(\lambda x.xx)(\lambda x.xx)$.

Proof. By structural induction on M, the interesting case being:

$$
\begin{array}{rl}
(NP) & {}^{<}\triangleq \mu\alpha.\langle N^{<}|\tilde{\mu}x.\langle M^{>}|x\cdot\alpha\rangle\rangle \\
& \to_{\tilde{\mu}} \mu\alpha.\langle M^{>}|N^{>}\cdot\alpha\rangle \\
& \triangleq (NP)^{>}
\end{array}
$$

□

This is a bit disappointing since the CBN-encoding turns out to be more refined than the CBV-encoding, breaking the nice symmetry.

On the other hand, we will show here that $\mathcal{X}$ fully preserves this symmetry. Of course, by Theorem 6.5, we also have

Lemma 8.2 $[\![M^{<}]\!]^{\lambda\mu\tilde{\mu}}_{\beta} \to^{*} [\![M^{>}]\!]^{\lambda\mu\tilde{\mu}}_{\beta}$

But we now show that we can take the simplest translation (i.e. $[\![\cdot^{>}]\!]^{\lambda\mu\tilde{\mu}}_{\beta}$) for both cases CBV and CBN and that the evaluation strategies of $\mathcal{X}$ reflect the distinction between them.

We assume the reader to be familiar with the λ-calculus [5]. We can define the direct encoding of the λ-calculus into $\mathcal{X}$:

Definition 8.3 *Let* $[\![M]\!]^{\mathrm{N}}_{\alpha} = [\![M^{>}]\!]^{\lambda\mu\tilde{\mu}}_{\alpha}$*, in other words:*

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

Notice that this is the encoding as defined in Definition 4.2.

Definition 8.4 (Curry types for the λ-calculus)
The type assignment rules for the Curry Type Assignment system for λ-calculus are:

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

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

We can now show that typeability is preserved by $\llbracket \cdot \rrbracket^{\mathrm{N}}_{\alpha}$:

Theorem 8.5 *If* $\Gamma \vdash_{\lambda} M : A$, *then* $\llbracket M \rrbracket^{\mathrm{N}}_{\alpha} :\cdot \Gamma \vdash \alpha{:}A$.

So, when encoding the Call-by-Value λ-calculus, we also use the $\llbracket \cdot \rrbracket^{\mathrm{N}}_{\alpha}$ interpretation. And, in contrast to $\lambda\mu\tilde{\mu}$, we can get an accurate interpretation of the Call-by-Value λ-calculus into $\mathcal{X}$ by using the CBV strategy, which we can reformulate as:

$$
\begin{array}{l}
(\textit{bend}\text{-L}) : M\widehat{\alpha} \dagger \widehat{x}N \;\rightarrow\; M\widehat{\alpha} \not\swarrow \widehat{x}N, \textit{ if } M \textit{ does not introduce } \alpha \\
(\textit{bend}\text{-R}) : M\widehat{\alpha} \dagger \widehat{x}N \;\rightarrow\; M\widehat{\alpha} \nwarrow \widehat{x}N, \\
\qquad\qquad \textit{if } M \textit{ introduces } \alpha \textit{ and } N \textit{ does not introduce } x
\end{array}
$$

If M introduces α and N introduces x, the logical rules apply. (This makes the calculus deterministic, but that is OK, since we want to simulate CBV reduction.)

Now notice that when M is a value (i.e. either a variable or an abstraction) then $\llbracket M \rrbracket^{\mathrm{N}}_{\alpha}$ introduces α. In fact, M is a value if and only if $\llbracket M \rrbracket^{\mathrm{N}}_{\alpha}$ introduces α. Then $\llbracket M \rrbracket^{\mathrm{N}}_{\alpha}\widehat{\alpha} \dagger \widehat{x}\llbracket N \rrbracket^{\mathrm{N}}_{\beta}$ cannot be reduced by rule $(\textit{bend}\text{-L})$, but by either rule $(\textit{bend}\text{-R})$ or a logical rule. This enables the reduction

$$\llbracket M \rrbracket^{\mathrm{N}}_{\alpha}\widehat{\alpha} \dagger \widehat{x}\llbracket N \rrbracket^{\mathrm{N}}_{\beta} \rightarrow_{\mathrm{V}}{}^{*} \llbracket N[M/x] \rrbracket^{\mathrm{N}}_{\beta}.$$

So Call-by-Value reduction for the λ-calculus is respected by the interpretation function, using the CBV strategy.

Theorem 8.6 (Simulation of λ-calculus)

1. *If* $M \rightarrow_{\mathrm{V}} N$ *then* $\llbracket M \rrbracket^{\mathrm{N}}_{\gamma} \rightarrow_{\mathrm{V}}{}^{*} \llbracket N \rrbracket^{\mathrm{N}}_{\gamma}$.
2. *If* $M \rightarrow_{\mathrm{N}} N$ *then* $\llbracket M \rrbracket^{\mathrm{N}}_{\gamma} \rightarrow_{\mathrm{N}}{}^{*} \llbracket N \rrbracket^{\mathrm{N}}_{\gamma}$.

Proof. This is a direct corollary of Theorem 4.5, the simulation result for $\lambda\mathbf{x}$ presented in the Section 4. □

9 Infinite Computations

To strengthen the fact that we consider more than just those circuits that represent proofs, we will show an example of a non-terminating reduction sequence.

Example 9.1 (Reducing $\llbracket \Delta\Delta \rrbracket^{\mathrm{N}}_{\beta}$) *Notice that*

$$
\begin{array}{ll}
\llbracket xx \rrbracket^{\mathrm{N}}_{\alpha} & \triangleq \\
\llbracket x \rrbracket^{\mathrm{N}}_{\gamma}\widehat{\gamma} \dagger \widehat{z}(\llbracket x \rrbracket^{\mathrm{N}}_{\beta}\widehat{\beta}\,[z]\;\widehat{y}\langle y.\alpha\rangle) & \triangleq \\
\langle x.\gamma\rangle\widehat{\gamma} \dagger \widehat{z}(\langle x.\beta\rangle\widehat{\beta}\,[z]\;\widehat{y}\langle y.\alpha\rangle) & \rightarrow (\textit{med}) \\
\langle x.\beta\rangle\widehat{\beta}\,[x]\;\widehat{y}\langle y.\alpha\rangle &
\end{array}
$$

and that this reduction is unique and deterministic. So we can define the short-hand

$$
\begin{array}{rcl}
\llbracket xx \rrbracket^{\mathrm{N}}_{\alpha} & = & \langle x.\beta\rangle\widehat{\beta}\,[x]\;\widehat{y}\langle y.\alpha\rangle \\
\llbracket \lambda x.xx \rrbracket^{\mathrm{N}}_{\gamma} & = & \widehat{x}\llbracket xx \rrbracket^{\mathrm{N}}_{\alpha}\widehat{\alpha}{\cdot}\gamma \\
\llbracket \Delta\Delta \rrbracket^{\mathrm{N}}_{\beta} & = & \llbracket \lambda x.xx \rrbracket^{\mathrm{N}}_{\gamma}\widehat{\gamma} \dagger \widehat{z}(\llbracket \lambda x.xx \rrbracket^{\mathrm{N}}_{\gamma}\widehat{\gamma}\,[z]\;\widehat{y}\langle y.\beta\rangle)
\end{array}
$$

and notice that now a weld only appears in the interpretation of $\Delta\Delta$. Now this term reduces as in Figure 6.

An alternative reduction sequence is:

$$\begin{array}{ll}(\widehat{x}\,[\![xx]\!]^{\mathrm{N}}_{\alpha}\widehat{\alpha}\cdot\delta)\widehat{\delta}\,\dagger\,\widehat{z}([\![\lambda x.xx]\!]^{\mathrm{N}}_{\gamma}\widehat{\gamma}\;[z]\;\widehat{y}\langle y.\beta\rangle) & \rightarrow \text{(exp-med)}\\ ([\![\lambda x.xx]\!]^{\mathrm{N}}_{\gamma}\widehat{\gamma}\,\dagger\,\widehat{x}[\![xx]\!]^{\mathrm{N}}_{\alpha})\widehat{\alpha}\,\dagger\,\widehat{y}\langle y.\beta\rangle & \rightarrow \text{(2.11-2)}\\ [\![\lambda x.xx]\!]^{\mathrm{N}}_{\gamma}\widehat{\gamma}\,\dagger\,\widehat{x}[\![xx]\!]^{\mathrm{N}}_{\beta} & \end{array}$$

so this forms a small diamond. In fact, all renaming reductions can be postponed without affecting the end result.

10 From sequent calculus to $\mathcal{X}$

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

$$(ax):\frac{}{\Gamma, A\vdash A,\Delta}\qquad (R\rightarrow):\frac{\Gamma, A\vdash B,\Delta}{\Gamma\vdash A\rightarrow B,\Delta}$$

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

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

Another nice property of proof systems that is known as the *Curry-Howard correspondence* is the fact that one can associate a term with a proof such that propositions become types and proof reductions become term reductions (or computations in $\mathcal{X}$). This phenomenon was first discovered for the λ-calculus and its connection with intuitionistic logic was put in evidence. The Curry-Howard correspondence for $\mathcal{X}$ is with classical propositional calculus and is given through the sequent calculus described above. Proposition receives 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 $\alpha, \beta, \gamma, \ldots$

An example: a proof of Peirce's Law

Peirce's Law can be shown to be inhabited in $\lambda\mu\tilde{\mu}$ by the term $\lambda z.\mu\alpha.\langle z|(\lambda y.\mu\beta.\langle y|\alpha\rangle)\cdot\alpha\rangle$ [14] which is itself the reduction of the translation of the $\lambda\mu$-term given by Ong and Steward [23]. Using the translation function $[\![\cdot]\!]^{\lambda\mu\tilde{\mu}}$ on this term, we obtain the first term of Figure 7. Notice that the process obtained via translation has four welds: removing these produces the (in-side out) reduction sequence.

$$\begin{array}{rl}
\widehat{z}((\langle z.\epsilon\rangle\widehat{\epsilon}\dagger\widehat{x}((\widehat{y}(\underline{(\langle y.\rho\rangle\widehat{\rho}\dagger\widehat{u}\langle u.\alpha\rangle)\widehat{\beta}}\dagger\widehat{v}\langle v.\eta\rangle)\widehat{\eta}\cdot\phi)\widehat{\phi}\,[x]\ \widehat{w}\langle w.\alpha\rangle))\widehat{\alpha}\dagger\widehat{x}\langle x.\delta\rangle)\widehat{\delta}\cdot\gamma & \rightarrow \\
\widehat{z}((\langle z.\epsilon\rangle\widehat{\epsilon}\dagger\widehat{x}((\widehat{y}\underline{(\langle y.\alpha\rangle\widehat{\beta}\dagger\widehat{v}\langle v.\eta\rangle)\widehat{\eta}}\cdot\phi)\widehat{\phi}\,[x]\ \widehat{w}\langle w.\alpha\rangle))\widehat{\alpha}\dagger\widehat{x}\langle x.\delta\rangle)\widehat{\delta}\cdot\gamma & \rightarrow \\
\widehat{z}(\underline{(\langle z.\epsilon\rangle\widehat{\epsilon}\dagger\widehat{x}((\widehat{y}\langle y.\alpha\rangle\widehat{\eta}\cdot\phi)\widehat{\phi}\,[x]\ \widehat{w}\langle w.\alpha\rangle))\widehat{\alpha}}\dagger\widehat{x}\langle x.\delta\rangle)\widehat{\delta}\cdot\gamma & \rightarrow \\
\underline{\widehat{z}(((\widehat{y}\langle y.\alpha\rangle\widehat{\eta}\cdot\phi)\widehat{\phi}\,[z]\ \widehat{w}\langle w.\alpha\rangle)\widehat{\alpha}\dagger\widehat{x}\langle x.\delta\rangle)\widehat{\delta}}\cdot\gamma & \rightarrow \\
\widehat{z}((\widehat{y}\langle y.\delta\rangle\widehat{\eta}\cdot\phi)\widehat{\phi}\,[z]\ \widehat{w}\langle w.\delta\rangle)\widehat{\delta}\cdot\gamma. &
\end{array}$$

Figure 7: Peirce's Law in $\mathcal{X}$.

Typing the contractum gives the derivation

$$\dfrac{\dfrac{\dfrac{\overline{\langle y.\delta\rangle\ :\cdot\ y{:}A\vdash\delta{:}A,\eta{:}B}}{\widehat{y}\langle y.\delta\rangle\widehat{\eta}\cdot\phi\ :\cdot\ \vdash\phi{:}A{\rightarrow}B,\delta{:}A} \qquad \overline{\langle w.\delta\rangle\ :\cdot\ w{:}A\vdash\delta{:}A}}{(\widehat{y}\langle y.\delta\rangle\widehat{\eta}\cdot\phi)\widehat{\phi}\,[z]\ \widehat{w}\langle w.\delta\rangle\ :\cdot\ z{:}(A{\rightarrow}B){\rightarrow}A\vdash\delta{:}A}}{\widehat{z}((\widehat{y}\langle y.\delta\rangle\widehat{\eta}\cdot\phi)\widehat{\phi}\,[z]\ \widehat{w}\langle w.\delta\rangle)\widehat{\delta}\cdot\gamma\ :\cdot\ \vdash\gamma{:}((A{\rightarrow}B){\rightarrow}A){\rightarrow}A}$$

when we remove all circuit information from this derivation, we obtain exactly the simplest proof for Peirce's Law in Classical Logic.

$$\dfrac{\dfrac{\dfrac{\overline{A\vdash A,B}}{\vdash A{\rightarrow}B,A} \qquad \overline{A\vdash A}}{(A{\rightarrow}B){\rightarrow}A\vdash A}}{\vdash((A{\rightarrow}B){\rightarrow}A){\rightarrow}A}$$

11 Conclusions and future work

This study is the first step of many researches. We want to focus in two directions: $\mathcal{X}$ as a language for designing circuit, especially with calculi like π-calculus, decidable typing in $\mathcal{X}$ and intersection types.

Acknowledgements We would like to thank Alexander Summers, Daniel Hirschkoff, Dragisa Zunic, Harry Mairson, Jayshan Raghunandan, Luca Cardelli, Luca Roversi, Maria Grazia Vigliotti, Nobuko Yoshida, and Simona Ronchi della Rocca for many fruitful discussion on the topic of this paper.

References

[1] M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. L´evy. Explicit substitutions. *Journal of Functional Programming*, 1(4):375–416, 1991.

[2] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. *Compilers: Principles, Techniques and Tools*. Addison-Wesley, 1988.

[3] A. W. Appel and T. Jim. Continuation-passing, closure-passing style. In *Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 293–302. ACM Press, 1989.

[4] F. Barbanera and S. Berardi. A symmetric lambda calculus for classical program extraction. *Information and Computation*, 125(2):103–117, 1996.

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

[6] H. P. Barendregt and S. Ghilezan. Lambda terms for natural deduction, sequent calculus and cut-elimination. *Journal of Functional Porgramming*, 10(1):121–134, 2000.

[7] R. Bloo and K.H. Rose. Preservation of strong normalisation in named lambda calculi with explicit substitution and garbage collection. In *CSN'95 – Computer Science in the Netherlands*, pages 62–72, 1995.

[8] P. Borovansky, C. Kirchner, H. Kirchner, P.E. Moreau, and C. Ringeissen:. An overview of elan. In C. and H. Kirchner, editors, *Proceedings of the 2nd International Workshop on Rewriting Logic and its Applications*, Pont-A-Mousson, France, September 1998.

[9] Manuel Clavel, Francisco Dur´an, Steven Eker, Patrick Lincoln, Narciso Mart´ı-Oliet, Jos´e Meseguer, and Carolyn Talcott. The maude 2.0 system. In Robert Nieuwenhuis, editor, *Rewriting Techniques and Applications (RTA 2003)*, number 2706 in Lecture Notes in Computer Science, pages 76–87. Springer-Verlag, June 2003.

[10] 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.

[11] Vincent Danos, Jean-Baptiste Joinet, and Harold Schellinx. Computational isomorphisms in classical logic (extended abstract). *Electronic Notes in Theoretical Computer Science*, 3, 1996.

[12] Vincent Danos, Jean-Baptiste Joinet, and Harold Schellinx. A new deconstructive logic: Linear logic. *The Journal of Symbolic Logic*, 62, 1997.

[13] A. G. Dragalin. *Mathematical Intuitionism: Introduction to Proof Theory*, volume 67 of *Translations of Mathematical Monographs*. American Mathematical Society, 1987.

[14] S. Ghilezan and P. Lescanne. Classical proofs, typed processes and intersection types. In S. Berardi, M. Coppo, and F. Damiani, editors, *TYPES'03*, 2004. forthcoming in Lecture Notes in Computer Science.

[15] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[16] J.-Y. Girard. A new constructive logic: classical logic. *Mathematical Structures in Computer Science*, 1(3):255–296, 1991.

[17] H. Herbelin. *Séquents qu'on calcule : de l'interprétation du calcul des séquents comme calcul de λ-termes et comme calcul de stratégies gagnantes*. Thèse d'universit´e, Universit´e Paris 7, Janvier 1995.

[18] I.Yarovsky, M.I.Aguilar, and M.T.W.Hearn. Computer simulation of peptide interaction with an rp-hplc sorbent. *J.Phys.Chem.B*, 101:10962–10970, 1997.

[19] St´ephane 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.

[20] St´ephane Lengrand, Pierre Lescanne, Dan Dougherty, Mariangiola Dezani-Ciancaglini, and Steffen van Bakel. Intersection types for explicit substitutions. *Information and Computation*, 189(1):17–42, 2004.

[21] P. Lescanne. From $\lambda\sigma$ to $\lambda\upsilon$, a journey through calculi of explicit substitutions. In Hans Boehm, editor, *Proceedings of the 21st Annual ACM Symposium on Principles Of Programming Languages, Portland (Or., USA)*, pages 60–69. ACM, 1994.

[22] Robin Milner. *Communicating and mobile systems: the π-calculus*. Cambridge University Press, 1999.

[23] 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.

[24] 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.

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

[26] Steffen van Bakel, Jayshan Raghunandan, and Alexander Summers. Term graphs, α-conversion, and principal types for $\mathcal{X}$. Submitted.

[27] 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.

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

[29] A. N. Whitehead and B. Russell. *Principia Mathematica to *56*. Cambridge Mathematical Library. Cambridge University Press, 2nd edition, 1997.