Erasure and Duplication in Classical Computation

D. Žunić & P. Lescanne

Ecole Normale Supérieure de Lyon
46, Allée d'Italie, 69364 Lyon Cedex 07, France
{pierre.lescanne, dragisa.zunic}@ens-lyon.fr

Abstract

We present the $*\mathcal{X}$-calculus, a linear model of computation, which has a direct Curry-Howard correspondence with classical logic. The logical setting is classical sequent calculus. $*\mathcal{X}$ introduces terms for explicit erasure and duplication, which correspond to weakening and contraction on the logical side. Furthermore, it is a calculus of explicit substitutions. Non-confluence and interface preservation are also some of the features. Since λ-calculus is easily represented, the untyped language is Turing-complete.

1. Introduction

The fundamental connection between logic and computation is known as the Curry-Howard correspondence. Originally it was used to refer to the connection between simply typed λ-calculus and intuitionistic logic. Later it became the "Curry-Howard paradigm", which means that this pattern of thinking was successfully applied to relate other calculi with their corresponding logics. Basically, it relates logical concepts term/reduction/type with computational concepts proof/normalization/proposition.

The first step in applying this paradigm to classical logic was done by T. Griffin [Gri90]. He noted the connection of classical logic with functional programming languages with control operators, which initiated the lively research in this area. After studying the computational content of classical natural deduction, for example the λ_μ-calculus of M. Parigot [Par92], people started considering classical sequent calculus, which exhibits various symmetries. There are several calculi that are based on sequent calculus formalism, for example one of the first was $\bar{\lambda}_{\mu\tilde{\mu}}$-calculus of P. L. Curien and H. Herbelin [CH00].

On the other side, although placed in the area of classical logic, our work has been strongly influenced by results in the field of intuitionistic logic and λ-calculus, namely by the results presented in [KL06] of D. Kesner and S. Lengrand. They designed the λ_{lxr}-calculus which is an extension of λ_x ([BR95]) with explicit operators for erasure and duplication.

$\mathcal{X}$-calculus before $*\mathcal{X}$-calculus The calculus we are presenting in this paper has its predecessor, it is the $\mathcal{X}$-calculus of S. van Bakel, P. Lescanne and S. Lengrand [vBLL05]. The origin of the language $\mathcal{X}$ lies in the notations for the sequent calculus (variant G3 of [Kle52]) as presented first by C. Urban [UB99, Urb00] and later studied in relation with $\bar{\lambda}_{\mu\tilde{\mu}}$-calculus by S. Lengrand [Len03] via the calculus there called λ_ϵ. Another paper [LZ06] presents $*\mathcal{X}$ from the point of view of diagrams, whereas this paper insists mostly on the term-calculus and its connection with sequent calculus. Moreover it compares $*\mathcal{X}$ with $\mathcal{X}$.

***$\mathcal{X}$-calculus** The *$\mathcal{X}$-calculus can be seen as an extension of $\mathcal{X}$-calculus. It has four extra sintactic elements, two for erasers and two for duplicators. Its logical setting is the formulation of classical sequent system known as G2 [Kle52]. It is a variant of Gentzen's LK [Gen35] where structural rules are explicitly given, contexts are lists of formulas and inference rules are given in the context-splitting style. Thus as the reader may assume, the new syntactic elements - eraser and duplicator, have an intuitive and direct logical interpretation as weakening and contraction. We also introduce the notion of *linearity* in *$\mathcal{X}$.

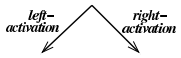
Substitution Our calculus is basically a calculus of explicit substitutions. A lot of research has been done on explicit substitutions, mostly in the context of λ-calculi, for example [ACCL91, Les94, BR95]. The main idea was to design calculi where the substitution will have a syntactic representation, unlike in λ-calculus where it is a meta-operation.

The main difference regarding explicit substitution, when we speak in the context of *$\mathcal{X}$ and classical logic, lies in the fact that here it can be applied to both sides, left or right. In λ_x we know that, once it is created, explicit substitution $\langle x = Q\rangle$ can be distributed only to the left-hand side

$$(\lambda x.P)Q \xrightarrow{\mathrm{B}} P\langle x = Q\rangle$$

whilst in *$\mathcal{X}$ it is not apriori decided which term will became substitution, the one on the left, or the one on the right. This has a simple syntactic representation

$$P\widehat{\alpha} \dagger \widehat{x}Q$$



$$P\widehat{\alpha} \nearrow \widehat{x}Q \qquad P\widehat{\alpha} \nwarrow \widehat{x}Q$$

where in one case Q plays the role of explicit substitution (more precisely: $[\widehat{\alpha} \nearrow \widehat{x}Q]$) and in the other case it is P (or more precisely: $[P\widehat{\alpha} \nwarrow \widehat{x}]$).

Duplication and Erasure The existence of an explicit term for duplication enables us to define a reduction engine which does not duplicate a term when there is no need. For comparison, in $\mathcal{X}$-calculus we always duplicate a term. This is due to the fact that the duplicator is hidden, which means that we do not know if it is really there. Since we do not want to miss any, we always have to assume it implicitly exists.

Similarly, the existence of a term called eraser, allows us to perform the cancellation i.e. erasing when there is no need to propagate the term further. In $\mathcal{X}$-calculus, terms that were supposed to be canceled, were propagated until they reach the level of ground terms (where the propagation must end) and only then would they get canceled.

Structure of the Paper The rest of the paper is organized as follows: in *section 2* we present the calculus itself, with the notion of linearity; *section 3* defines a typing system revealing the correspondence with sequential classical logic; *section 4* describes a relation between $\mathcal{X}$-calculus and *$\mathcal{X}$; *section 5* presents the encoding of λ-calculus and finally *section 6* is a conclusion with future work.

2. The Calculus

In this section we give the syntax of untyped *$\mathcal{X}$-calculus as well as the set of reduction rules.

2.1. The Syntax

Terms are built from *names*. This concept differs essentially from one applied in λ-calculus, where the notion of a *variable* is essential. The difference lies in the fact that a variable can be substituted by an arbitrary term, while a name can be only *renamed* (substituted by another name). In our calculus the renaming is explicit, which means that it is expressed within the language itself. The reader will notice the presence of hats on some names. This notation has been borrowed from *Principia Mathematica* [WR25] and is used to denote the binding of a name.

An interesting feature of $^*\mathcal{X}$ is that the same construction, e.g., an exporter, an importer, a cut and a duplicator, binds two names. A duplicator binds either two in-names or two out-names, whereas the other operators bind an in-name and an out-name. Moreover the names that are bound can belong to the same term, or to two different terms. For example, *importer* and *cut*, which are dyadic operations, bind an out-name in a term on the left and an in-name in a term on the right.

Definition 1 ($^*\mathcal{X}$-terms) *The terms of the $^*\mathcal{X}$-calculus are given by the syntax represented by Figure 1, where x, y... range over the infinite set of* in-names *and α, β... over the infinite set of* out-names.

P,Q	::=	$\langle x.\alpha\rangle$	*capsule*
	|	$\widehat{x}\,P\,\widehat{\beta}\cdot\alpha$	*exporter*
	|	$P\,\widehat{\alpha}\;[x]\;\widehat{y}\,Q$	*importer*
	|	$P\widehat{\alpha}\;\dagger\;\widehat{x}Q$	*cut*
	|	$x\odot P$	*left-eraser*
	|	$P\odot\alpha$	*right-eraser*
	|	$z{<}^{\widehat{x}}_{\widehat{y}}\langle P]$	*left-duplicator*
	|	$[P\rangle^{\widehat{\alpha}}_{\widehat{\beta}}{>}\gamma$	*right-duplicator*

Figure 1: The Syntax

The Components of the Syntax

CAPSULE : $\langle x.\alpha\rangle$ It is the basic element for constructing terms, i.e. a ground term, as it can not be decomposed. It has one in-name x and one out-name α.

EXPORTER : $(\widehat{y}\,P\,\widehat{\beta}\cdot\alpha)$ It is a term which binds two names, an in-name y and an out-name β of the subterm P. Together they represent a functionality which is made available on the new output α.

IMPORTER : $(P\,\widehat{\beta}\;[x]\;\widehat{y}\,Q)$ It consists of two subterms P and Q and it binds an out-name β of the first term, namely P, and an in-name x of the second, namely Q. These two names are meant to be connected to each other. Since their type can be different, a "mediation" is necessary. This is made through an in-name y of functional type. Eventually an importer is aimed to interact with an exporter. Thus, an importer is the counterpart of an exporter. The reduction rule that describes the interaction between an exporter and an importer is in fact the basic computational step in $^*\mathcal{X}$-calculus.

CUT : $(P\widehat{\alpha}\;\dagger\;\widehat{x}Q)$ It represents a direct connection between the two subterms P and Q. Cut connects α in P with x in Q. Actually, the terminology "cut" looks odd for a "connection" (the terminology comes from the proof theory). Here, "cut" $\dagger$ denotes something that links, or attempts to connect, but never something that separates or cuts. We could say, it 'cuts' through the structure of a term in order to 'link' two names (i.e., two terms over those two names).

ERASERS (Left and Right) : $x \odot P$ and $P \odot \alpha$ The first one explicitly adds the in-name (which, in the linear case, does not occur in P) to be free in the expression $x \odot P$. Similarly $P \odot \alpha$ adds the free out-name to a term P. These operators corresponds to the inference rules called *left and right weakening* in the sequent calculus.

DUPLICATORS (Left and Right) : $x{<}^{\widehat{x_1}}_{\widehat{x_2}}\langle P]$ and $[P\rangle^{\widehat{\alpha_1}}_{\widehat{\alpha_2}}{>}\alpha$ As we will deal only with linear terms, these operators are a way to simulate multiple occurrences of names in a term. More precisely $x{<}^{\widehat{x_1}}_{\widehat{x_2}}\langle P]$ allows x to pretend it would occur twice in P, but the linearity of P is preserved. Actually x is duplicated into x_1 and x_2.

Free Names The set of free names, i.e. free in-names and free out-names is defined by Figure 2. We write fn to denote the *set* of names and f_{in} and f_{on} to denote the *sets* of in-names and out-names, respectively. Thus we have $fn(P) = f_{in}(P) \cup f_{on}(P)$. A name which is not free is called a *bound name*.

S	$f_{in}(S)$	$f_{on}(S)$
$\langle x.\alpha\rangle$	x	α
$\widehat{x}\,P\,\widehat{\beta}\cdot\alpha$	$f_{in}(P)\backslash\{x\}$	$(f_{on}(P)\backslash\{\beta\})\cup\{\alpha\}$
$P\,\widehat{\alpha}\,[x]\,\widehat{y}\,Q$	$f_{in}(P)\cup(f_{in}(Q)\backslash\{y\})\cup\{x\}$	$(f_{on}(P)\backslash\{\alpha\})\cup f_{on}(Q)$
$P\widehat{\alpha}\,\dagger\,\widehat{x}Q$	$f_{in}(P)\cup(f_{in}(Q)\backslash\{x\})$	$(f_{on}(P)\backslash\{\alpha\})\cup f_{on}(Q)$
$x \odot P$	$f_{in}(P)\cup\{x\}$	$f_{on}(P)$
$P \odot \alpha$	$f_{in}(P)$	$f_{on}(P)\cup\{\alpha\}$
$x{<}^{\widehat{x_1}}_{\widehat{x_2}}\langle P]$	$(f_{in}(P)\backslash\{x_1,\ x_2\})\cup\{x\}$	$f_{on}(P)$
$[P\rangle^{\widehat{\alpha_1}}_{\widehat{\alpha_2}}{>}\alpha$	$f_{in}(P)$	$(f_{on}(P)\backslash\{\alpha_1,\ \alpha_2\})\cup\{\alpha\}$

Figure 2: Free Names

We are assuming that the terms are equal up to the α-conversion, which means up to the renaming of bound in-names or out-names.

We adopt a convention known as Barendregt's convention on variables, which states that free and bound names are always considered to be distinct. In $^*\mathcal{X}$-calculus it will be a convention on names: *"A name is never both, free and bound, in a term"*.

If we wish to think of sets $f_n(P)$, $f_{in}(P)$ and $f_{on}(P)$ as lists, we will use the notation $\Phi_n(P)$ (or simply Φ), $\Phi_{in}(P)$ and $\Phi_{on}(P)$, respectively. The simpler way would be to write Φ^P_n, Φ^P_{in} and Φ^P_{on}. If we wish to exclude the name, say out-name z, from the list, we write $\Phi^P_{on}\backslash z$

Indexing In what follows, free names will be renamed by indexing. Let i be an index (a natural), P a term and Φ a set of free names occurring in P. Then $ind(P, \Phi, i)$ means for example: a name x belonging to Φ is replaced in P by x_i and a name α belonging to Φ is replaced in P by α_i. To avoid losing linearity, we assume that we start with non-indexed names. The way we use it, indexing preserves linearity.

When Φ_{in} is a list of in-names, we write $\Phi_{in,i}$ for the same list of names indexed by i and similarly with Φ_{on} for out-names.

Linearity A term is linear if it satisfies the following:

- Every name has at most one free occurrence and

- Every binder does bind an occurrence of a name (does bind something)

In our calculus we consider only linear terms. Although the syntax in general produces non-linear terms, it appears that they have a linear representation. Take for example the term $\langle x.\alpha\rangle \odot \alpha$, which is not linear since the out-name α has two free occurrences. It can be represented in $^*\mathcal{X}$ by the term $[\langle x.\alpha_1\rangle \odot \alpha_2\rangle^{\widehat{a_1}}_{\widehat{a_2}} > a$. The other cause of non-linearity is when we have a binder which binds no name, for example in a term $\widehat{x}\,\langle x.\alpha\rangle\,\widehat{\beta}\cdot\gamma$. The problematic binder is $\widehat{\beta}$, and this is solved by using $^*\mathcal{X}$-term $\widehat{x}\,(\langle x.\alpha\rangle \odot \beta)\,\widehat{\beta}\cdot\gamma$. According to this, what enables us to stay linear is the presence of terms for weakening and contraction.

For the formal definition of linear terms, see Figure 3.

$$\frac{}{\langle x.\alpha\rangle \;\; linear}$$

$$\frac{P \;\; linear,\;\; x \in f_{in}(P), \beta \in f_{on}(P),\;\; \alpha \notin f_{on}(P)}{\widehat{x}\,P\,\widehat{\beta}\cdot\alpha \;\; linear}$$

$$\frac{P,Q \;\; linear,\;\; \alpha \in f_{on}(P),\; x \in f_{in}(Q),\;\; y \notin f_{in}(P,Q),\;\; f_n(P) \cap f_n(Q) = \emptyset}{P\,\widehat{\alpha}\,[y]\,\widehat{x}\,Q \;\; linear}$$

$$\frac{P,Q \;\; linear,\;\; \alpha \in f_{on}(P),\; x \in f_{in}(Q),\;\; f_n(P) \cap f_n(Q) = \emptyset}{P\widehat{\alpha} \dagger \widehat{x}Q \;\; linear}$$

$$\frac{P \;\; linear,\;\; x \notin f_{in}(P)}{x \odot P \;\; linear} \qquad \frac{P \;\; linear,\;\; \alpha \notin f_{on}(P)}{P \odot \alpha \;\; linear}$$

$$\frac{P \;\; linear,\;\; x,y \in f_{in}(P),\;\; z \notin f_{in}(P)}{z{<}^{\widehat{x}}_{\widehat{y}}\langle P] \;\; linear} \qquad \frac{P \;\; linear,\;\; \alpha,\beta \in f_{on}(P),\;\; \gamma \notin f_{on}(P)}{[P\rangle^{\widehat{\alpha}}_{\widehat{\beta}} > \gamma \;\; linear}$$

Figure 3: Linear Terms

2.2. Reduction Rules

In this section we define the reduction relation $\xrightarrow{*\mathcal{X}}$. As for the cut-elimination, this system has many rules. Therefore we found it convenient to split them into five classes, which are themselves split into symmetrical subclasses "left" and "right", with the exception of logical rules.

We are also assuming some simple equivalences on terms, coming from the commutativity of weakenings, as well as from associativity and commutativity of formulas in contractions.

Reduction rules are divided into

- Activation rules (left and right)
- Propagation rules (left and right)
- Actions (left and right)
- Deactivation rules (left and right)
- Logical rules

2.2.1. Activation Rules

Reduction rules proceed towards eliminating cuts in terms. In the process of reducing there are situations when one has to *choose* the side where cuts will be propagated. This choice will be explicitly represented by bending the cut towards either the left or the right. A bended cut is called an *active cut*. The syntax will be extendend so that it can show the direction of activation.

Definition 2 (Active Cuts) *The syntax is extended with the two* active cuts*:*

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

Remark The terms $P\widehat{\alpha} \nearrow \widehat{x}Q$ and $P\widehat{\alpha} \nwarrow \widehat{x}Q$ are essentially different. Perhaps this is best illustrated by Lafont's example [GLT89]. Take $P = M \odot \alpha$ and $Q = x \odot N$, where M and N are arbitrary terms. Then we have

$$\begin{array}{lcl} (M \odot \alpha)\widehat{\alpha} \nearrow \widehat{x}(x \odot N) & \rightarrow & \Phi^N_{in} \odot M \odot \Phi^N_{on} \\ (M \odot \alpha)\widehat{\alpha} \nwarrow \widehat{x}(x \odot N) & \rightarrow & \Phi^M_{in} \odot N \odot \Phi^M_{on} \end{array}$$

On the other hand the actual activation is defined by the reduction rules. See Figure 4. In general, a cut can be activated in both ways.

$$\begin{array}{lcl} (L-activation) & : & P\widehat{\alpha} \dagger \widehat{x}Q \rightarrow P\widehat{\alpha} \nearrow \widehat{x}Q, \text{ with } P \neq \widehat{y}\,P'\,\widehat{\beta}\cdot\alpha \text{ and } P \neq \langle y.\alpha\rangle \\ (R-activation) & : & P\widehat{\alpha} \dagger \widehat{x}Q \rightarrow P\widehat{\alpha} \nwarrow \widehat{x}Q, \text{ with } Q \neq Q'\,\widehat{\beta}\,[x]\,\widehat{y}\,Q'' \text{ and } Q \neq \langle x.\beta\rangle \end{array}$$

Figure 4: Activation Rules

The existence of this *choice* has a consequence that $^*\mathcal{X}$ is a *non-confluent* calculus. But if one is interested in the property of confluence, it is easily achievable by giving priority to either left or right-activation (in the case when both are possible). We can denote those reduction relations as $\xrightarrow{^*\mathcal{X}_l}$ and $\xrightarrow{^*\mathcal{X}_r}$. Thus we obtain two calculi which are both confluent.

2.2.2. Propagation Rules

Propagation rules describe how cut is being propagated through the structure of terms. This is a step by step propagation (reduction rules "describe" propagation). It is significant to note that propagation of a cut over another inactive cut is enabled, which allows the possibility to elegantly represent β-reduction. The rules are divided into "left" and "right" symmetric groups, see Figures 5 and 8.

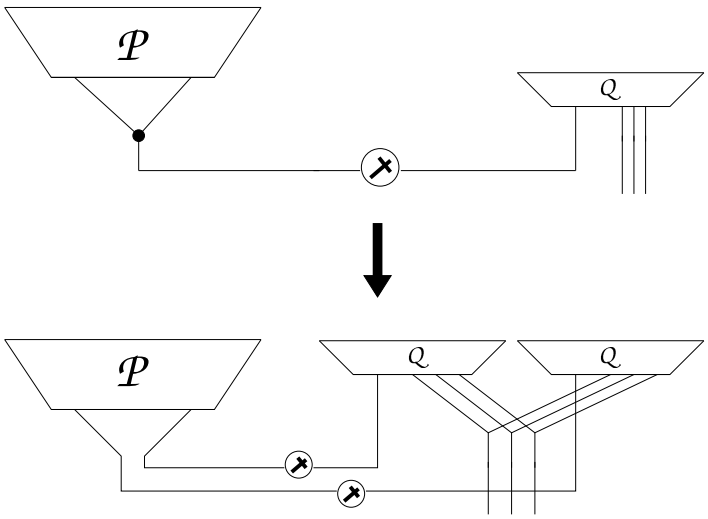
Observe for example the first reduction rule ($exp\nearrow - prop$) from the left group; it shows how an active cut (or if you wish, an explicit substitution $[\widehat{\beta} \nearrow \widehat{y}R]$) enters from the left through an export, and then stops at its immediate subterm.

2.2.3. Actions

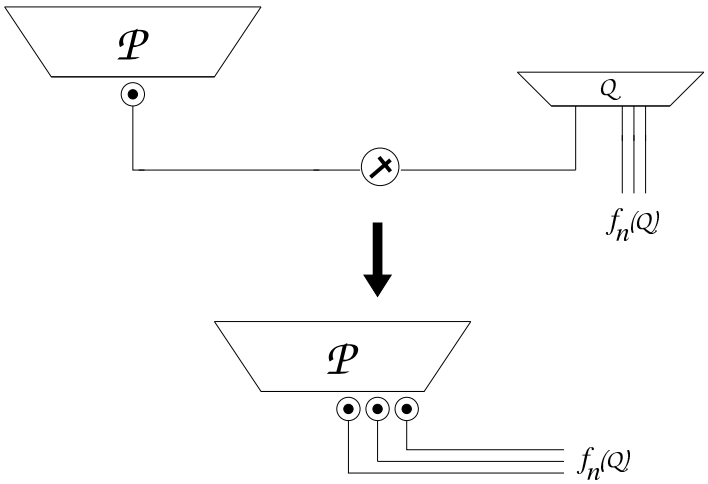
Actions are reduction rules which specify erasure and duplication. They refer to the situation when the cut which is being propagated meets an eraser or a duplicator. See Figures 6 and 9.

The *erasure* of a term is performed in such a way that no free names are lost in the computation. They are "attached" to the remaining term.In the process of computation we also have *duplication*. The duplication of terms is performed in such a way that we do not allow the same free name to occur more times. Here we are using indexing to keep duplicated free names distinct. According to this, actions preserve the *linearity* of terms. We used abbreviations to present the rules for erasure and duplication.

The syntax of those rules may look complicated to the reader, therefore we give a graphical illustration (which can simultaneously be viewed as a proof-transformation). The action of duplicating the term can be represented in the following way:



The same stands for the action of erasing a term, it becomes more clear when shown in the illustration below:



2.2.4. Deactivation Rules

Once the cuts are activated they are performing various activities, they can be propagated, erased, duplicated, but also *deactivated*. The important point in the reduction procedure is that they can not be deactivated at will. It is the deactivation group of rules that specifies when a cut is deactivated. See Figures 7 and 10. Only when a cut is in the situation that it can not be activated again, we can apply reductions from the group of logical rules.

Notice that the interplay between the reductions specifying the activation and the ones specifying the deactivation can not cause infinite reduction sequences of the form *act*, *deact*, *act*, *deact* ..., due to side conditions.

This page contains figures specifying left-directed reductions.

$$
\begin{array}{lcll}
(exp\nearrow - prop) & : & (\widehat{x}\,P\widehat{\gamma}\cdot\alpha)\widehat{\beta}\nearrow\widehat{y}R & \rightarrow\ \widehat{x}\,(P\widehat{\beta}\nearrow\widehat{y}R)\widehat{\gamma}\cdot\alpha, \quad \alpha\neq\beta \\
(imp\nearrow - prop_1) & : & (P\,\widehat{\alpha}\,[x]\,\widehat{z}\,Q)\widehat{\beta}\nearrow\widehat{y}R & \rightarrow\ (P\widehat{\beta}\nearrow\widehat{y}R)\,\widehat{\alpha}\,[x]\,\widehat{z}Q, \quad \beta\in f_{on}(P) \\
(imp\nearrow - prop_2) & : & (P\,\widehat{\alpha}\,[x]\,\widehat{z}\,Q)\widehat{\beta}\nearrow\widehat{y}R & \rightarrow\ P\,\widehat{\alpha}\,[x]\,\widehat{z}\,(Q\widehat{\beta}\nearrow\widehat{y}R), \quad \beta\in f_{on}(Q) \\
(cut(caps)\nearrow - prop) & : & (P\widehat{\alpha}\dagger\widehat{x}\langle x.\beta\rangle)\widehat{\beta}\nearrow\widehat{y}R & \rightarrow\ P\widehat{\alpha}\dagger\widehat{y}R \\
(cut\nearrow - prop_1) & : & (P\widehat{\alpha}\dagger\widehat{x}Q)\widehat{\beta}\nearrow\widehat{y}R & \rightarrow\ (P\widehat{\beta}\nearrow\widehat{y}R)\widehat{\alpha}\dagger\widehat{x}Q, \quad \beta\in f_{on}(P),\ q\neq\langle x.\beta\rangle \\
(cut\nearrow - prop_2) & : & (P\widehat{\alpha}\dagger\widehat{x}Q)\widehat{\beta}\nearrow\widehat{y}R & \rightarrow\ P\widehat{\alpha}\dagger\widehat{x}(Q\widehat{\beta}\nearrow\widehat{y}R), \quad \beta\in f_{on}(Q),\ q\neq\langle x.\beta\rangle \\
(L\text{–}eras\nearrow - prop) & : & (x\odot P)\widehat{\beta}\nearrow\widehat{y}R & \rightarrow\ x\odot(P\widehat{\beta}\nearrow\widehat{y}R) \\
(R\text{–}eras\nearrow - prop) & : & (P\odot\alpha)\widehat{\beta}\nearrow\widehat{y}R & \rightarrow\ (P\widehat{\beta}\nearrow\widehat{y}R)\odot\alpha, \quad \alpha\neq\beta \\
(L\text{–}dupl\nearrow - prop) & : & (x{<}\widehat{\frac{x_1}{x_2}}\langle P]\rangle)\widehat{\beta}\nearrow\widehat{y}R & \rightarrow\ x{<}\widehat{\frac{x_1}{x_2}}\langle P\widehat{\beta}\nearrow\widehat{y}R] \\
(R\text{–}dupl\nearrow - prop) & : & ([P\rangle\widehat{\frac{\alpha_1}{\alpha_2}}{>}\alpha)\widehat{\beta}\nearrow\widehat{y}R & \rightarrow\ [P\widehat{\beta}\nearrow\widehat{y}R\rangle\widehat{\frac{\alpha_1}{\alpha_2}}{>}\alpha, \quad \alpha\neq\beta
\end{array}
$$

Figure 5: Left Propagation

$$
\begin{array}{lcll}
(\nearrow erasure) & : & (P\odot\beta)\widehat{\beta}\nearrow\widehat{y}R & \rightarrow\ \Phi_{in}\odot P\odot\Phi_{on} \\
 & & & \Phi_{in}=f_{in}(R)\backslash\{y\},\ \Phi_{on}=f_{on}(R) \\
(\nearrow duplication) & : & ([P\rangle\widehat{\frac{\beta_1}{\beta_2}}{>}\beta)\widehat{\beta}\nearrow\widehat{y}R & \rightarrow\ \Phi_{in}{<}\widehat{\frac{\Phi_{in,1}}{\Phi_{in,2}}}\langle(P\widehat{\beta_1}\nearrow\widehat{y_1}R_1)\widehat{\beta_2}\nearrow\widehat{y_2}R_2\rangle\widehat{\frac{\Phi_{on,1}}{\Phi_{on,2}}}{>}\Phi_{on} \\
 & & & \Phi_{in}=f_{in}(R)\setminus\{y\},\ \Phi_{on}=f_{on}(R) \\
 & & & R_1=ind(R,\Phi_{in}\cup\Phi_{on},1) \qquad R_2=ind(R,\Phi_{in}\cup\Phi_{on},2)
\end{array}
$$

Figure 6: Outname Actions

$$
\begin{array}{lcll}
(cap\nearrow - deactivation) & : & \langle x.\beta\rangle\widehat{\beta}\nearrow\widehat{y}R & \rightarrow\ \langle x.\beta\rangle\widehat{\beta}\dagger\widehat{y}R \\
(exp\nearrow - deactivation) & : & (\widehat{x}\,P\widehat{\gamma}\cdot\beta)\widehat{\beta}\nearrow\widehat{y}R & \rightarrow\ (\widehat{x}\,P\widehat{\gamma}\cdot\beta)\widehat{\beta}\dagger\widehat{y}R
\end{array}
$$

Figure 7: Left Deactivation

This page contains figures describing right-directed reductions.

$$
\begin{array}{lcll}
(\nwarrow exp-prop) & : & P\widehat{\alpha} \nwarrow \widehat{x}(\widehat{y}\,Q\,\widehat{\beta}\cdot\gamma) & \rightarrow \quad \widehat{y}\,(P\widehat{\alpha} \nwarrow \widehat{x}Q)\,\widehat{\beta}\cdot\gamma \\
(\nwarrow imp-prop_1) & : & P\widehat{\alpha} \nwarrow \widehat{x}(Q\,\widehat{\beta}\,[y]\,\widehat{z}\,R) & \rightarrow \quad (P\widehat{\alpha} \nwarrow \widehat{x}Q)\,\widehat{\beta}\,[y]\,\widehat{z}\,R, \quad x \in f_{in}(Q) \\
(\nwarrow imp-prop_2) & : & P\widehat{\alpha} \nwarrow \widehat{x}(Q\,\widehat{\beta}\,[y]\,\widehat{z}\,R) & \rightarrow \quad Q\,\widehat{\beta}\,[y]\,\widehat{z}\,(P\widehat{\alpha} \nwarrow \widehat{x}R), \quad x \in f_{in}(R) \\
(\nwarrow cut(caps)-prop) & : & P\widehat{\alpha} \nwarrow \widehat{x}(\langle x.\beta\rangle\widehat{\beta} \dagger \widehat{y}R) & \rightarrow \quad P\widehat{\alpha} \dagger \widehat{y}R \\
(\nwarrow cut-prop_1) & : & P\widehat{\alpha} \nwarrow \widehat{x}(Q\widehat{\beta} \dagger \widehat{y}R) & \rightarrow \quad (P\widehat{\alpha} \nwarrow \widehat{x}Q)\widehat{\beta} \dagger \widehat{y}R, \quad x \in f_{in}(Q),\ Q \neq \langle x.\beta\rangle \\
(\nwarrow cut-prop_2) & : & P\widehat{\alpha} \nwarrow \widehat{x}(Q\widehat{\beta} \dagger \widehat{y}R) & \rightarrow \quad Q\widehat{\beta} \dagger \widehat{y}(P\widehat{\alpha} \nwarrow \widehat{x}R), \quad x \in f_{in}(R),\ Q \neq \langle x.\beta\rangle \\
(\nwarrow L\text{–}eras-prop) & : & P\widehat{\alpha} \nwarrow \widehat{x}(y \odot Q) & \rightarrow \quad y \odot (P\widehat{\alpha} \nwarrow \widehat{x}Q), \quad x \neq y \\
(\nwarrow R\text{–}eras-prop) & : & P\widehat{\alpha} \nwarrow \widehat{x}(Q \odot \beta) & \rightarrow \quad (P\widehat{\alpha} \nwarrow \widehat{x}Q) \odot \beta \\
(\nwarrow L\text{–}dupl-prop) & : & P\widehat{\alpha} \nwarrow \widehat{x}(y{<}^{\widehat{y_1}}_{\widehat{y_2}}\langle Q]) & \rightarrow \quad y{<}^{\widehat{y_1}}_{\widehat{y_2}}\langle P\widehat{\alpha} \nwarrow \widehat{x}Q], \quad x \neq y \\
(\nwarrow R\text{–}dupl-prop) & : & P\widehat{\alpha} \nwarrow \widehat{x}([Q\rangle^{\widehat{\beta_1}}_{\widehat{\beta_2}}{>}\beta) & \rightarrow \quad [P\widehat{\alpha} \nwarrow \widehat{x}Q\rangle^{\widehat{\beta_1}}_{\widehat{\beta_2}}{>}\beta
\end{array}
$$

Figure 8: Right Propagation

$$
\begin{array}{lcll}
(\nwarrow erasure) & : & P\widehat{\alpha} \nwarrow \widehat{x}(x \odot Q) & \rightarrow \quad \Phi_{in} \odot Q \odot \Phi_{on} \\
 & & & \quad \Phi_{in} = f_{in}(P),\ \ \Phi_{on} = f_{on}(P)\backslash\{\alpha\} \\
(\nwarrow duplication) & : & P\widehat{\alpha} \nwarrow \widehat{x}(x{<}^{\widehat{x_1}}_{\widehat{x_2}}\langle Q]) & \rightarrow \quad \Phi_{in} {<}^{\widehat{\Phi_{in,1}}}_{\widehat{\Phi_{in,2}}}\langle P_2\widehat{\alpha_2}\nwarrow\widehat{x_2}(P_1\widehat{\alpha_1}\nwarrow\widehat{x_1}Q\rangle^{\widehat{\Phi_{on,1}}}_{\widehat{\Phi_{on,2}}}){>}\,\Phi_{on} \\
 & & & \quad \Phi_{in} = f_{in}(P),\ \ \Phi_{on} = f_{on}(P) \setminus \{\alpha\} \\
 & & & \quad P_1 = ind(P, \Phi_{in} \cup \Phi_{on}, 1) \qquad P_2 = ind(P, \Phi_{in} \cup \Phi_{on}, 2)
\end{array}
$$

Figure 9: Inname Actions

$$
\begin{array}{lcll}
(\nwarrow cap-deactivation) & : & P\widehat{\alpha} \nwarrow \widehat{x}\langle x.\beta\rangle & \rightarrow \quad P\widehat{\alpha} \dagger \widehat{x}\langle x.\beta\rangle \\
(\nwarrow imp-deactivation) & : & P\widehat{\alpha} \nwarrow \widehat{x}(Q\,\widehat{\beta}\,[x]\,\widehat{y}\,R) & \rightarrow \quad P\widehat{\alpha} \dagger \widehat{x}(Q\,\widehat{\beta}\,[x]\,\widehat{y}\,R)
\end{array}
$$

Figure 10: Right Deactivation

2.2.5. Logical Rules

Logical rules describe the cut-elimination between two terms which both introduce names involved in a cut, i.e. they refer to cuts which can not be activated. See Figure 11.

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

Figure 11: Logical Rules

The first three logical rules specify the renaming. The third logical rule represents the basic computational step. It describes the direct interaction between an exporter and an importer, which results in inserting the subterm of an exporter between the two subterms of an importer (notice that $y, \beta \in P$).

The reduction system enjoys some nice properties, as given by the following lemma.

Lemma 1 (The basic properties of $\xrightarrow{*\mathcal{X}}$)

1. *The reductions preserve the set of free names: If $P \xrightarrow{*\mathcal{X}} Q$ then $f_n(P) = f_n(Q)$ (Interface preservation [Laf95]).*
2. *The reductions preserve linearity: If P is linear and $P \xrightarrow{*\mathcal{X}} Q$ then Q is linear.*

Proof. By treating each reduction rule, the proof proceeds by induction on the structure of terms.

3. Type System

In this framework, types are arrow types. Given a set T of basic types, a type is given by

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

The type assignment of a term is given in the form $T \vdots \Gamma \vdash \Delta$, where Γ is a set[1] of type declarations for in-names, e.g., $\Gamma \equiv x:A,\ y:B$ and Δ is a set of declarations for out-names, e.g., $\Delta \equiv \alpha:A,\ \beta:A \rightarrow B,\ \gamma:C$. The type system for $*\mathcal{X}$ is given in Figure 12.

Classical Sequent Calculus If we remove the term-decoration in the type system of Figure 12, we get the sequent system for classical logic. See Figure 13. The sequent system we consider is very similar to Kleene's G1 variant [Kle52].

The reader will notice that this system is an *implicational fragment* of *classical sequent calculus* in the *context-splitting* style. An important point to note is the presence of *explicit structural rules* (weakening and contraction).

[1] In fact Γ and Δ are multisets, but since all names are distinct we can treat them as sets

$$\frac{}{\langle x.\alpha\rangle \vdots \cdot \;\; x:A \vdash \alpha:A}\ \textit{(capsule)}$$

$$\frac{P \vdots \cdot \;\; \Gamma \vdash \alpha:A,\Delta \qquad Q \vdots \cdot \;\; \Gamma', y:B \vdash \Delta'}{P\,\widehat{\alpha}\,[x]\,\widehat{y}Q \vdots \cdot \;\; \Gamma,\Gamma', x:A \to B \vdash \Delta,\Delta'}\ \textit{(importer)} \qquad \frac{P \vdots \cdot \;\; \Gamma, x:A \vdash \alpha:B,\Delta}{\widehat{x}\,P\,\widehat{\alpha}\cdot\beta \vdots \cdot \;\; \Gamma \vdash \beta:A \to B,\Delta}\ \textit{(exporter)}$$

$$\frac{P \vdots \cdot \;\; \Gamma \vdash \alpha:A,\Delta \qquad Q : \Gamma', x:A \vdash \Delta'}{P\widehat{\alpha} \dagger \widehat{x}Q \vdots \cdot \;\; \Gamma,\Gamma' \vdash \Delta,\Delta'}\ \textit{(cut)}$$

$$\frac{P \vdots \cdot \;\; \Gamma \vdash \Delta}{x \odot P \vdots \cdot \;\; \Gamma, x:A \vdash \Delta}\ \textit{(L-eraser)} \qquad \frac{P \vdots \cdot \;\; \Gamma \vdash \Delta}{P \odot \alpha \vdots \cdot \;\; \Gamma \vdash \alpha:A,\Delta}\ \textit{(R-eraser)}$$

$$\frac{P \vdots \cdot \;\; \Gamma, x:A, y:A \vdash \Delta}{z{<}\frac{\widehat{x}}{\widehat{y}}\langle P\rangle \vdots \cdot \;\; \Gamma, z:A \vdash \Delta}\ \textit{(L-duplicator)} \qquad \frac{P \vdots \cdot \;\; \Gamma \vdash \alpha:A, \beta:A,\Delta}{[P\rangle^{\widehat{\alpha}}_{\widehat{\beta}}{>}\gamma \vdots \cdot \;\; \Gamma \vdash \gamma:A,\Delta}\ \textit{(R-duplicator)}$$

Figure 12: Type System

$$\frac{}{A \vdash A}\ \textit{(axiom)}$$

$$\frac{\Gamma \vdash A,\Delta \qquad \Gamma', B \vdash \Delta'}{\Gamma,\Gamma', A \to B \vdash \Delta,\Delta'}\ \textit{(L}\to\textit{)} \qquad \frac{\Gamma, A \vdash B,\Delta}{\Gamma \vdash A \to B,\Delta}\ \textit{(R}\to\textit{)}$$

$$\frac{\Gamma \vdash A,\Delta \qquad \Gamma', A \vdash \Delta'}{\Gamma,\Gamma' \vdash \Delta,\Delta'}\ \textit{(cut)}$$

$$\frac{\Gamma \vdash \Delta}{\Gamma, A \vdash \Delta}\ \textit{(L-weakening)} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash A,\Delta}\ \textit{(R-weakening)}$$

$$\frac{\Gamma, A, A \vdash \Delta}{\Gamma, A \vdash \Delta}\ \textit{(L-contraction)} \qquad \frac{\Gamma \vdash A, A,\Delta}{\Gamma \vdash A,\Delta}\ \textit{(R-contraction)}$$

Figure 13: Classical Sequent Calculus

• *Implicational* means that only arrows (or implications) occur to form types (or propositions).
• *Classical* means that the types (or the propositions) are those of the implicational fragment of classical propositional logic.
• *Context splitting* means that in the rules with two premises, namely $(L \to)$ and (cut), contexts are split.

If we want to see reductions of $^*\mathcal{X}$ as proof-transformations, the subject reduction property is essential.

Theorem 1 (Subject Reduction) *If* $S \vdots \cdot \;\; \Gamma \vdash \Delta$ *and* $S \xrightarrow{^*\mathcal{X}} S'$, *then* $S' \vdots \cdot \;\; \Gamma \vdash \Delta$

Proof. Due to a lack of space we will give the proof for some of the non-trivial cases of reduction rules. First we write the typing derivations corresponding to S and after, the derivations corresponding to S'.

- Take $(\widehat{y}\,P\,\widehat{\beta}\cdot\alpha)\widehat{\alpha} \dagger \widehat{x}\langle x.\gamma\rangle \;\xrightarrow{^*\mathcal{X}}\; \widehat{y}\,P\,\widehat{\beta}\cdot\gamma$. On one hand we have

$$\dfrac{\dfrac{\boxed{\mathrm{P}} \vdots\ \Gamma, y : A \vdash \beta : B, \Delta}{\boxed{\widehat{y}\,P\,\widehat{\beta}\cdot\alpha} \vdots\ \Gamma \vdash \alpha : A \to B, \Delta}\,(\to R) \qquad \dfrac{}{\boxed{\langle x.\gamma\rangle} \vdots\ x : A \to B \vdash \gamma : A \to B}\,(ax)}{\boxed{(\widehat{y}\,P\,\widehat{\beta}\cdot\alpha)\widehat{\alpha} \dagger \widehat{x}\langle x.\gamma\rangle} \vdots\ \Gamma \vdash \gamma : A \to B, \Delta}\,(cut)$$

On the other hand,

$$\dfrac{\boxed{\mathrm{P}} \vdots\ \Gamma, y : A \vdash \beta : B, \Delta}{\boxed{\widehat{y}\,P\,\widehat{\beta}\cdot\gamma} \vdots\ \Gamma \vdash \gamma : A \to B, \Delta}\,(\to R)$$

• Take $(\widehat{y}\,P\,\widehat{\beta}\cdot\alpha)\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\gamma}\ [x]\ \widehat{z}\,R) \xrightarrow{*\mathcal{X}} Q\widehat{\gamma} \dagger \widehat{y}(P\widehat{\beta} \dagger \widehat{z}R)$, with $y, \beta \in P$. On one hand we have

$$\dfrac{\dfrac{\boxed{\mathrm{P}} \vdots\ \Gamma, y : A \vdash \beta : B, \Delta}{\boxed{\widehat{y}\,P\,\widehat{\beta}\cdot\alpha} \vdots\ \Gamma \vdash \alpha : A \to B, \Delta}\,(\to R) \qquad \dfrac{\boxed{\mathrm{Q}} \vdots\ \Gamma' \vdash \gamma : A, \Delta' \qquad \boxed{\mathrm{R}} \vdots\ \Gamma'', z : B \vdash \Delta''}{\boxed{Q\widehat{\gamma}\ [x]\ \widehat{z}\,R} \vdots\ \Gamma', \Gamma'', x : A \to B \vdash \Delta', \Delta''}\,(\to L)}{\boxed{(\widehat{y}\,P\,\widehat{\beta}\cdot\alpha)\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\gamma}\ [x]\ \widehat{z}\,R)} \vdots\ \Gamma, \Gamma', \Gamma'' \vdash \Delta, \Delta', \Delta''}\,(cut)$$

On the other hand,

$$\dfrac{\boxed{\mathrm{Q}} \vdots\ \Gamma' \vdash \gamma : A, \Delta' \qquad \dfrac{\boxed{\mathrm{P}} \vdots\ \Gamma, y : A \vdash \beta : B, \Delta \qquad \boxed{\mathrm{R}} \vdots\ \Gamma'', z : B \vdash \Delta''}{\boxed{P\widehat{\beta} \dagger \widehat{z}R} \vdots\ \Gamma, \Gamma'', y : A \vdash \Delta, \Delta''}\,(\to L)}{\boxed{Q\widehat{\gamma} \dagger \widehat{y}(P\widehat{\beta} \dagger \widehat{z}R)} \vdots\ \Gamma, \Gamma', \Gamma'' \vdash \Delta, \Delta', \Delta''}\,(cut)$$

• Take $(P \odot \beta)\widehat{\beta} \nearrow \widehat{y}R \xrightarrow{*\mathcal{X}} \Phi^R_{in} \odot P \odot \Phi^R_{on}$. On one hand we have

$$\dfrac{\dfrac{\boxed{\mathrm{P}} \vdots\ \Gamma \vdash \Delta}{\boxed{P \odot \beta} \vdots\ \Gamma \vdash \beta : B, \Delta}\,(weak\text{-}R) \qquad \boxed{\mathrm{R}} \vdots\ \Gamma', y : B \vdash \Delta'}{\boxed{(P \odot \beta)\widehat{\beta} \nearrow \widehat{y}R} \vdots\ \Gamma, \Gamma' \vdash \Delta, \Delta'}\,(cut)$$

On the other hand,

$$\dfrac{\dfrac{\boxed{\mathrm{P}} \vdots\ \Gamma \vdash \Delta}{\vdots}\,(weak)}{\boxed{\Phi^R_{in} \odot P \odot \Phi^R_{on}} \vdots\ \Gamma, \Gamma' \vdash \Delta, \Delta'}\,(weak)$$

4. $\mathcal{X}$ vs $^*\mathcal{X}$

Perhaps the best explanation of $^*\mathcal{X}$-calculus is through an illustrative example. The reader will also be given a chance to compare the same example in the framework of $\mathcal{X}$-calculus. We will prove the Peirce law, a proposition which can not be proven in intuitionistic logic.

An Example We give the proof of Peirce law $((A \to B) \to A) \to A$ in our logical setting - LK with explicit weakening and contraction. Through Curry-Howard correspondence we can witness how the $^*\mathcal{X}$-term for Peirce law is being built:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\langle x.\alpha_1\rangle \therefore\ x : A \vdash \alpha_1 : A}\,(capsule)}{\langle x.\alpha_1\rangle \odot \beta \therefore\ x : A \vdash \alpha_1 : A, \beta : B}\,(R\text{-}eraser)}{\widehat{x}\,(\langle x.\alpha_1\rangle \odot \beta)\,\widehat{\beta}\cdot\gamma \therefore\ \vdash \alpha_1 : A, \gamma : A \to B}\,(exporter) \quad \dfrac{}{\langle y.\alpha_2\rangle \therefore\ y : A \vdash \alpha_2 : A}\,(capsule)}{(\widehat{x}\,(\langle x.\alpha_1\rangle \odot \beta)\,\widehat{\beta}\cdot\gamma)\,\widehat{\gamma}\ [z]\ \widehat{y}\,\langle y.\alpha_2\rangle \therefore\ z : (A \to B) \to A \vdash \alpha_1 : A, \alpha_2 : A}\,(importer)}{[(\widehat{x}\,(\langle x.\alpha_1\rangle \odot \beta)\,\widehat{\beta}\cdot\gamma)\,\widehat{\gamma}\ [z]\ \widehat{y}\,\langle y.\alpha_2\rangle\rangle^{\widehat{\alpha_1}}_{\widehat{\alpha_2}} > \alpha \therefore\ z : (A \to B) \to A \vdash \alpha : A}\,(R\text{-}duplicator)}{\widehat{z}\,([(\widehat{x}\,(\langle x.\alpha_1\rangle \odot \beta)\,\widehat{\beta}\cdot\gamma)\,\widehat{\gamma}\ [z]\ \widehat{y}\,\langle y.\alpha_2\rangle\rangle^{\widehat{\alpha_1}}_{\widehat{\alpha_2}} > \alpha)\,\widehat{\alpha}\cdot\delta \therefore\ \vdash \delta : ((A \to B) \to A) \to A}\,(exporter)}$$

When we remove the term−annotation we get:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A}\,(axiom)}{A \vdash A, B}\,(R\text{-}weakening)}{\vdash A, A \to B}\,(\to R) \quad \dfrac{}{A \vdash A}\,(axiom)}{(A \to B) \to A \vdash A, A}\,(\to L)}{(A \to B) \to A \vdash A}\,(R\text{-}contraction)}{\vdash ((A \to B) \to A) \to A}\,(\to R)$$

In order to allow reader to compare, we give the same proof corresponding to $\mathcal{X}$-calculus. See Figure 14.

5. Encoding λ-Calculus

The calculus we presented can be used to encode various calculi. Here we give the encoding of λ-calculus which proves that untyped $^*\mathcal{X}$ is Turing-complete.

In order to present the encoding in a more elegant way we define the operation $\circledcirc$ which adds erasers where necessary:

$$x \circledcirc [\![M]\!]_\alpha = \begin{cases} x \odot [\![M]\!]_\alpha, & x \notin fv(M) \\ [\![M]\!]_\alpha, & x \in fv(M) \end{cases}$$

Notice that M is a λ-calculus term and that, in this context, x is a variable of the λ-calculus (hence the use of fv for "free variables"). The same letters are used for both, the variables of λ-calculus and the in-names of $^*\mathcal{X}$-calculus. Indeed, they correspond to each other in the sense that a free variable in λ-calculus becomes a free in-name in $^*\mathcal{X}$, after the encoding.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\langle x.\alpha\rangle \vdots\ x:A \vdash \alpha:A,\beta:B}\,(capsule)}{\widehat{x}\,\langle x.\alpha\rangle\,\widehat{\beta}\cdot\gamma \vdots\ \vdash \alpha:A,\gamma:A\to B}\,(exporter) \quad \dfrac{}{\langle y.\alpha\rangle \vdots\ y:A \vdash \alpha:A}\,(capsule)}{(\widehat{x}\,\langle x.\alpha\rangle\,\widehat{\beta}\cdot\gamma)\widehat{\gamma}\,[z]\,\widehat{y}\,\langle y.\alpha\rangle \vdots\ z:(A\to B)\to A \vdash \alpha:A}\,(importer)}{\widehat{z}\,((\widehat{x}\,\langle x.\alpha\rangle\,\widehat{\beta}\cdot\gamma)\widehat{\gamma}\,[z]\,\widehat{y}\,\langle y.\alpha\rangle)\,\widehat{\alpha}\cdot\delta \vdots\ \vdash \delta:((A\to B)\to A)\to A}\,(exporter)$$

When we remove the term-annotation we get:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A,B}\,(axiom)}{\vdash A,A\to B}\,(\to R) \quad \dfrac{}{A \vdash A}\,(axiom)}{(A\to B)\to A \vdash A}\,(\to L)}{\vdash ((A\to B)\to A)\to A}\,(\to R)$$

Figure 14: Peirce Law in $\mathcal{X}$

Definition 3 (Encoding λ-calculus) *The encoding of λ-calculus terms in $^*\mathcal{X}$-calculus is defined as follows:*

$$\begin{array}{rcl}
[\![x]\!]_\alpha & := & \langle x.\alpha\rangle \\
[\![\lambda x.M]\!]_\alpha & := & \widehat{x}\,(x \circledcirc [\![M]\!]_\beta)\,\widehat{\beta}\cdot\alpha \\
[\![MN]\!]_\alpha & := & \Phi_{in}{<}\overbrace{\Phi_{in,1}}^{\overbrace{\Phi_{in,2}}}\langle \underline{M}\widehat{\gamma} \dagger \widehat{x}\underline{N}],
\end{array}$$

where
$\Phi_{in} = fv(M) \cap fv(N) \setminus \{x\}$
$\underline{M} = ind([\![M]\!]_\gamma, \Phi_{in}, 1)$
$\underline{N} = ind([\![N]\!]_\beta, \Phi_{in}, 2)\,\widehat{\beta}\ [x]\ \widehat{y}\,\langle y.\alpha\rangle$

Lemma 2 (Faithfulness) *If $M \xrightarrow{\beta} N$, then $[\![M]\!]_\alpha \xrightarrow{^*\mathcal{X}} (f_{in}([\![M]\!]_\alpha) \setminus f_{in}([\![N]\!]_\alpha)) \odot [\![N]\!]_\alpha$.*

For the simple case, when $f_{in}([\![M]\!]_\alpha) \setminus f_{in}([\![N]\!]_\alpha) = \emptyset$, the property becomes $[\![M]\!]_\alpha \xrightarrow{^*\mathcal{X}} [\![N]\!]_\alpha$.

6. Conclusion and Future Work

This work is a link in a chain of works in the field of computational representation of classical logic. The calculus we presented has many desirable properties. The fact that $^*\mathcal{X}$ is a linear model of computation for classical logic, can be used to study classical computation in two (or more) dimensions, i.e. it can lead to a design of some type of diagrammatic calculus for functional programming.

References

[ACCL91] M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy. Explicit substitutions. *Journal of Functional Programming*, 1(4):375–416, 1991.

[BR95] 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.

[Gen35] G. Gentzen. Untersuchungen über das logische Schließen. *Mathematische Zeitschrift*, 39:176–210, 405–431, 1935.

[CH00] P.-L. Curien and H. 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.

[GLT89] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*, volume 7 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1989. available on `http://www.cs.man.ac.uk/~pt/stable/Proofs+Types.html`.

[Gri90] T. Griffin. A formulae-as-types notion of control. In *Proceedings of the 17th Annual ACM Symposium on Principles Of Programming Languages, Orlando (Fla., USA)*, pages 47–58, 1990.

[KL06] D. Kesner and S. Lengrand. Explicit operators for λ-calculus. *Information and Computation*, 2006. extended version of a communication at RTA-05, to be published.

[Kle52] S. C. Kleene. *Introduction to Metamathematics*. Number 1 in Bibliotheca mathematica. North-Holland, 1952. Revised edition, Wolters-Noordhoff, 1971.

[Laf95] Y. Lafont. From proof nets to interaction nets. In J.-Y. Girard, Y. Lafont, and L. Regnier, editors, *Advances in Linear Logic*, pages 225–247. Cambridge University Press, 1995.

[Len03] S. 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.

[Les94] 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.

[LZ06] P. Lescanne and D. Žunić. $*\mathcal{X}$: a diagrammatic calculus with a classical fragrance. To be submitted.

[Par92] 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.

[UB99] C. Urban and G. M. Bierman. Strong normalisation of cut-elimination in classical logic. In *Typed Lambda Calculus and Applications*, volume 1581 of *Lecture Notes in Computer Science*, pages 365–380. Springer-Verlag, 1999.

[Urb00] C. Urban. *Classical Logic and Computation*. PhD thesis, University of Cambridge, October 2000.

[vBLL05] S. van Bakel, S. Lengrand, and P. Lescanne. The language $\mathcal{X}$: circuits, computations and classical logic. In Mario Coppo, Elena Lodi, and G. Michele Pinna, editors, *Proceedings of Ninth Italian Conference on Theoretical Computer Science (ICTCS'05), Siena, Italy*, volume 3701 of *Lecture Notes in Computer Science*, pages 81–96. Springer-Verlag, 2005.

[WR25] A N. Whitehead and B. Russell. *Principia Mathematica*. Cambridge University Press, 2nd edition, 1925.