A congruence relation for restructuring classical terms

Dragiša Žunić[1] and Pierre Lescanne[2]

[1] Carnegie Mellon University in Qatar

[2] Laboratoire de l'Informatique du Parallélisme, LIP
Ecole Normale Supérieure de Lyon, France

Abstract. We present a congruence relation on sequent-style classical proofs which identifies proofs up to trivial rule permutation. The study is performed in the framework of ${}^*\mathcal{X}$ calculus which provides a Curry-Howard correspondence for classical logic (with explicit structural rules) ensuring that proofs can be seen as terms and proof transformation as computation. Congruence equations provide an explicit account for classifying proofs (terms) which are syntactically different but essentially the same. We are able to prove that there is always a representative of a congruence class which enables computation at the top level.

1 Introduction

For a long time it was considered not possible to give constructive semantics to classical logic, but only to intuitionistic and linear logic. Recent works have shown that this is actually possible if one gives up on the principle that the computational semantics is a confluent rewrite system.

In this work we study the essence of classical proofs by means of specifying its unessential content. Namely we present a list of congruence rules specifying which syntactically different proofs/terms should be considered the same. Moreover, this congruence relation enables us to pick a representative of a congruence class so that the cut elimination can continue at the top level. This is done in the framework of ${}^*\mathcal{X}$ calculus - a higher order rewrite system designed to provide a Curry-Howard correspondence with the sequent system $G1$ [13] for classical logic (presented in Fig. 2). This system is characterized by the presence of structural rules weakening and contraction and it is very close to the original formulation by Gentzen [5].

2 Related work

The first computational interpretation of classical logic relying on sequents was presented by Curien and Herbelin in [4] and in [8], while a direct correspondence with a standard sequent formulation of classical logic, with the intent of capturing its computational content in full richness (although with implicit structural

rules), was presented by Urban in [14] and by Bierman and Urban in [15]. This research further lead to the formulation of $\mathcal{X}$ calculus [11, 2] which was a base to introduce explicit control over erasure and duplication by featuring explicit structural rules weakening and contraction, yielding the ${}^*\mathcal{X}$ calculus [18, 6]. This kind of conservative calculus extension had been studied in the intuitionistic framework by Kesner and Lengrand [9]. Indeed there is a certain analogy in going from λx [1] (featuring explicit substitution) towards λlxr [9] (explicit substitution, erasure and duplication) in the natural deduction setting, and going from $\mathcal{X}$ to ${}^*\mathcal{X}$ in the classical sequent calculus setting.

3 The calculus ${}^*\mathcal{X}$

Intuitively when we speak about ${}^*\mathcal{X}$-terms we speak about classical proofs formalized in the sequent calculus with explicit structural rules weakening and contraction. Terms are built from *names*. This concept differs essentially from the one applied in λ-calculus, where *variable* is the basic notion. The difference lies in the fact that a variable can be substituted by an arbitrary term, while a name can be only *renamed* (that is, substituted by another name). In our calculus the renaming is explicit, which means that it is expressed within the language itself and is not defined in the meta-theory. The notation of using hats over names has been borrowed from *Principia Mathematica* [17] and is used to denote the binding of a name.

Definition 1. *The* syntax *of ${}^*\mathcal{X}$-calculus[3] is presented in Fig. 1, where $x, y, z...$ range over an infinite set of in-names and $\alpha, \beta, \gamma...$ range over an infinite set of out-names.*

$P, Q ::= \langle x.\alpha \rangle$		*capsule*	(axiom rule)
	$\mid \widehat{x}\, P\, \widehat{\beta} \cdot \alpha$	*exporter*	(ritht arrow-introduction)
	$\mid P\, \widehat{\alpha}\, [x]\, \widehat{y}\, Q$	*importer*	(left arrow-introduction)
	$\mid P\widehat{\alpha} \dagger \widehat{x}Q$	*cut*	(cut)
	$\mid x \odot P$	*left-eraser*	(left weakening)
	$\mid P \odot \alpha$	*right-eraser*	(right weakening)
	$\mid z{<}^{\widehat{x}}_{\widehat{y}}\langle P]$	*left-duplicator*	(left contraction)
	$\mid [P\rangle^{\widehat{\alpha}}_{\widehat{\beta}}{>}\gamma$	*right-duplicator*	(right contraction)

Fig. 1. The syntax of ${}^*\mathcal{X}$

Although we distinguish two categories of names (in-names and out-names) they will usually be referred to simply as *names*. There are eight constructors introduced in the syntax, whose names suggest their computational role.

[3] We consider only the implicational fragment here.

Definition 2. *We define* free names *and* principal names *of a term R, written $fn(R)$ and $pn(R)$, respectively, in the following way:*

R	$fn(R)$	$pn(R)$
$\langle x.\alpha \rangle$	x, α	x, α
$\widehat{x}\, P\, \widehat{\beta} \cdot \alpha$	$fn(P) \backslash \{x, \beta\} \cup \alpha$	α
$P\, \widehat{\alpha}\ [x]\ \widehat{y}\, Q$	$fn(P) \cup fn(Q) \backslash \{\alpha, y\} \cup x$	x
$P \widehat{\alpha} \dagger \widehat{x} Q$	$fn(P) \cup fn(Q) \backslash \{\alpha, x\}$	*none*
$x \odot P$	$fn(P) \cup x$	x
$P \odot \alpha$	$fn(P) \cup \alpha$	α
$x < \frac{\widehat{x_1}}{\widehat{x_2}} \langle P]$	$fn(P) \backslash \{x_1, x_2\} \cup x$	x
$[P \rangle \frac{\widehat{\alpha_1}}{\widehat{\alpha_2}} > \alpha$	$fn(P) \backslash \{\alpha_1, \alpha_2\} \cup \alpha$	α

A name which is not free is called a *bound* name. The notion of a principal name describes a free name that appears at the top level in the structure of term. We will use P^{α} and Q^{x} to emphasize that P and Q have α and x as principal names, respectively.

Our consideration of principal names is motivated by the nature of cut operation, which always refers to a pair of free names (an out-name and an in-name) - one in each corresponding term - and binds them. If these names are at the top level (accessible) then the cut elimination proceeds in one of its essential steps. However this research is concerned by static aspects only (dealing with proof structure), but it most certainly attempts to lay the foundation for revealing the computational essence as well.

In the ${}^{*}\mathcal{X}$ calculus we consider only linear terms. We say that a term is *linear* if every name has at most one free occurrence and every binder does bind an actual occurrence of a name (and therefore only one). We use the standard definition of *linear terms*, see [18] on page 43. We will specify the notions of *context*, *subterm* and *immediate subterm* in order to efficiently work with the structure of terms.

Definition 3. *The* contexts *are formally defined as follows:*

$$
\begin{array}{lcllll}
C\{\ \} & ::= & \{\ \} & | & \widehat{x}\,\{\ \}\,\widehat{\beta} \cdot \alpha \\
 & | & \{\ \}\,\widehat{\alpha}\ [x]\ \widehat{y}\, Q & | & P\,\widehat{\alpha}\ [x]\ \widehat{y}\,\{\ \} \\
 & | & \{\ \}\widehat{\alpha} \dagger \widehat{x} Q & | & P \widehat{\alpha} \dagger \widehat{x}\{\ \} \\
 & | & x \odot \{\ \} & | & \{\ \} \odot \alpha \\
 & | & z < \frac{\widehat{x}}{\widehat{y}} \langle \{\ \}] & | & [\{\ \} \rangle \frac{\widehat{\alpha}}{\widehat{\beta}} > \gamma \\
 & | & C\{C\{\ \}\} & &
\end{array}
$$

Informally, a context is a term with a hole which can accept another term. Therefore $C\{P\}$ denotes placing the term P in the context $C\{\ \}$.

Definition 4. *A term Q is a* subterm *of a term P, denoted as $Q \preccurlyeq P$ if there is a context $C\{\ \}$ such that $P = C\{Q\}$ (the symbol $=$ stands for syntactic equality.).*

It is easy to show that the *subterm* relation is reflexive, antisymmetric and transitive (i.e., is an order).

Definition 5. *A term Q is an* immediate subterm *of P if $P = C\{Q\}$ and $C\{\ \} \neq C'\{C''\{\ \}\},\ C\{\ \} \neq \{\ \}$.*

4 The type assignment system

Given a set T of basic types, a type is given by $A, B ::= A \in T \mid A \to B$. Expressions of the form $P :\cdot \Gamma \vdash \Delta$ are used to present the *type assignment*, where P is an ${}^*\mathcal{X}$ term and Γ, Δ are contexts whose domains consist of free in-names and out-names of P, respectively[4]. Comma in the expression Γ, Γ' stands for the set union.

Definition 6. *We say that a term P is* typable *if there exist contexts Γ and Δ such that $P :\cdot \Gamma \vdash \Delta$ is derivable in the system of rules given by Fig. 3.*

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

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

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

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

Fig. 2. Sequent system $G1$ for classical logic

Reduction rules are numerous as they capture the richness and complexity of classical cut elimination, and very fine-grained due to the presence of explicit terms for erasure and duplication. For details see [18].

5 The congruence relation

This section presents a congruence relation on terms, denoted $\equiv$, which is represented by a list of equations[5]. The congruence relation induced by these rules

[4] Technically contexts are sets of pairs (name, formula); if we forget about labels and consider only type, we are going back to Gentzen's classical system $G1$ (Fig. 2), where contexts are multisets of formulas.

[5] The list is partial due to limited space. A complete list of congruence rules can be found in [18].

$$\frac{}{\langle x.\alpha\rangle \ :\cdot\ x:A \vdash \alpha:A}\ (ax)$$

$$\frac{P \ :\cdot\ \Gamma \vdash \alpha:A,\Delta \qquad Q \ :\cdot\ \Gamma', y:B \vdash \Delta'}{P\,\widehat{\alpha}\ [x]\ \widehat{y}\,Q \ :\cdot\ \Gamma,\Gamma', x:A\to B \vdash \Delta,\Delta'}\ (L{\to}) \qquad \frac{P \ :\cdot\ \Gamma, x:A \vdash \alpha:B,\Delta}{\widehat{x}\,P\,\widehat{\alpha}\cdot\beta \ :\cdot\ \Gamma \vdash \beta:A\to B,\Delta}\ (R{\to})$$

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

$$\frac{P \ :\cdot\ \Gamma \vdash \Delta}{x \odot P \ :\cdot\ \Gamma, x:A \vdash \Delta}\ (weak\text{-}L) \qquad \frac{P \ :\cdot\ \Gamma \vdash \Delta}{P \odot \alpha \ :\cdot\ \Gamma \vdash \alpha:A,\Delta}\ (weak\text{-}R)$$

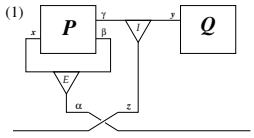
$$\frac{P \ :\cdot\ \Gamma, x:A, y:A \vdash \Delta}{z{<}^{\widehat{x}}_{\widehat{y}}\langle P] \ :\cdot\ \Gamma, z:A \vdash \Delta}\ (cont\text{-}L) \qquad \frac{P \ :\cdot\ \Gamma \vdash \alpha:A,\beta:A,\Delta}{[P\rangle^{\widehat{\alpha}}_{\widehat{\beta}}{>}\gamma \ :\cdot\ \Gamma \vdash \gamma:A,\Delta}\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

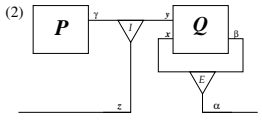
Fig. 3. The type system for ${}^{*}\mathcal{X}$

is a reflexive, symmetric and transitive relation closed under any context. These rules define explicitly and in an intuitive way which syntactically different terms should be considered the same. The key property of this congruence relation is that, given a cut term, it offers a representative of a congruence class so that the computation (cut elimination) continues at the top level (see Theorem 1).

The diagrammatic representation is assigned to every congruence rule to strengthen the reader's intuition (a formal study of the diagrammatic calculus, which can be derived from the term calculus presented here, is in the domain of future work). A label is assigned to every congruence rule, and thus each rule is declared as: $name : P \equiv Q$.

exporter-importer

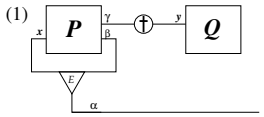


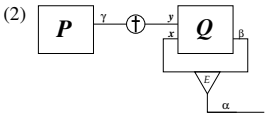


$ei1 \ : \ \widehat{x}\,(P\widehat{\gamma}\ [z]\ \widehat{y}\,Q)\,\widehat{\beta}\cdot\alpha \ \equiv \ (\widehat{x}\,P\,\widehat{\beta}\cdot\alpha)\,\widehat{\gamma}\ [z]\ \widehat{y}\,Q$ with $x,\beta \in fn(P)$

$ei2 \ : \ \widehat{x}\,(P\widehat{\gamma}\ [z]\ \widehat{y}\,Q)\,\widehat{\beta}\cdot\alpha \ \equiv \ P\widehat{\gamma}\ [z]\ \widehat{y}\,(\widehat{x}\,Q\,\widehat{\beta}\cdot\alpha)$ with $x,\beta \in fn(Q)$

exporter-cut



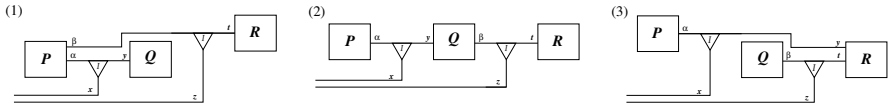


$ec1 \;:\; \widehat{x}\,(P\widehat{\gamma} \dagger \widehat{y}Q)\,\widehat{\beta}\cdot\alpha \;\equiv\; (\widehat{x}\,P\,\widehat{\beta}\cdot\alpha)\widehat{\gamma} \dagger \widehat{y}Q$ with $x, \beta \in fn(P)$

$ec2 \;:\; \widehat{x}\,(P\widehat{\gamma} \dagger \widehat{y}Q)\,\widehat{\beta}\cdot\alpha \;\equiv\; P\widehat{\gamma} \dagger \widehat{y}(\widehat{x}\,Q\,\widehat{\beta}\cdot\alpha)$ with $x, \beta \in fn(Q)$

---

importer-importer



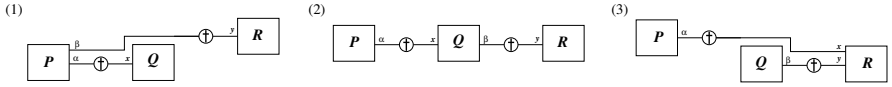
$ii1 \;:\; (P\,\widehat{\alpha}\,[x]\,\widehat{y}\,Q)\,\widehat{\beta}\,[z]\,\widehat{t}\,R \;\equiv\; (P\,\widehat{\beta}\,[z]\,\widehat{t}\,R)\,\widehat{\alpha}\,[x]\,\widehat{y}\,Q$ with $\alpha, \beta \in fn(P)$

$ii2 \;:\; (P\,\widehat{\alpha}\,[x]\,\widehat{y}\,Q)\,\widehat{\beta}\,[z]\,\widehat{t}\,R \;\equiv\; P\,\widehat{\alpha}\,[x]\,\widehat{y}\,(Q\,\widehat{\beta}\,[z]\,\widehat{t}\,R)$ with $y, \beta \in fn(Q)$

$ii3 \;:\; (Q\,\widehat{\beta}\,[z]\,\widehat{t}\,R) \;\equiv\; Q\,\widehat{\beta}\,[z]\,\widehat{t}\,(P\,\widehat{\alpha}\,[x]\,\widehat{y}\,R)$ with $y, t \in fn(R)$

---

cut-cut



$cc1 \;:\; (P\widehat{\alpha} \dagger \widehat{x}Q)\widehat{\beta} \dagger \widehat{y}R \;\equiv\; (P\widehat{\beta} \dagger \widehat{y}R)\widehat{\alpha} \dagger \widehat{x}Q$ with $\alpha, \beta \in fn(P)$

$cc2 \;:\; (P\widehat{\alpha} \dagger \widehat{x}Q)\widehat{\beta} \dagger \widehat{y}R \;\equiv\; P\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\beta} \dagger \widehat{y}R)$ with $x, \beta \in fn(Q)$

$cc3 \;:\; P\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\beta} \dagger \widehat{y}R) \;\equiv\; Q\widehat{\beta} \dagger \widehat{y}(P\widehat{\alpha} \dagger \widehat{x}R)$ with $x, y \in fn(R)$

---

cut-importer

$ci1 \;:\; (P\,\widehat{\alpha}\,[x]\,\widehat{y}\,Q)\widehat{\beta} \dagger \widehat{z}R \;\equiv\; (P\widehat{\beta} \dagger \widehat{z}R)\,\widehat{\alpha}\,[x]\,\widehat{y}\,Q$ with $\alpha, \beta \in fn(P)$

$ci2 \;:\; (P\,\widehat{\alpha}\,[x]\,\widehat{y}\,Q)\widehat{\beta} \dagger \widehat{z}R \;\equiv\; P\,\widehat{\alpha}\,[x]\,\widehat{y}\,(Q\widehat{\beta} \dagger \widehat{z}R)$ with $y, \beta \in fn(Q)$

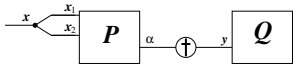
$ci3 \;:\; P\widehat{\alpha} \dagger \widehat{x}(Q\,\widehat{\beta}\,[y]\,\widehat{z}\,R) \;\equiv\; (P\widehat{\alpha} \dagger \widehat{x}Q)\,\widehat{\beta}\,[y]\,\widehat{z}\,R$ with $x, \beta \in fn(Q)$

$ci4 \;:\; P\widehat{\alpha} \dagger \widehat{x}(Q\,\widehat{\beta}\,[y]\,\widehat{z}\,R) \;\equiv\; Q\,\widehat{\beta}\,[y]\,\widehat{z}\,(P\widehat{\alpha} \dagger \widehat{x}R)$ with $x, z \in fn(R)$

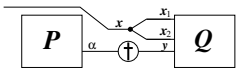
---

cut-contraction

(1)



(2)

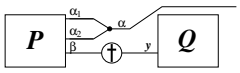


cc_t1 : $x<\widehat{\frac{x_1}{x_2}}\langle P\widehat{\alpha} \dagger \widehat{y}Q] \equiv (x<\widehat{\frac{x_1}{x_2}}\langle P])\widehat{\alpha} \dagger \widehat{y}Q$ with $x_1, x_2 \in fn(P)$

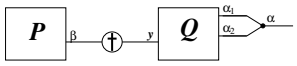
cc_t2 : $x<\widehat{\frac{x_1}{x_2}}\langle P\widehat{\alpha} \dagger \widehat{y}Q] \equiv P\widehat{\alpha} \dagger \widehat{y}(x<\widehat{\frac{x_1}{x_2}}\langle Q])$ with $x_1, x_2 \in fn(Q),\ y \neq x$

---

(3)



(4)



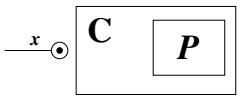
cc_t3 : $[P\widehat{\beta} \dagger \widehat{y}Q\rangle^{\widehat{\alpha_1}}_{\widehat{\alpha_2}}>\alpha \equiv ([P\rangle^{\widehat{\alpha_1}}_{\widehat{\alpha_2}}>\alpha)\widehat{\beta} \dagger \widehat{y}Q$ with $\alpha_1, \alpha_2 \in fn(P),\ \alpha \neq \beta$

cc_t4 : $[P\widehat{\beta} \dagger \widehat{y}Q\rangle^{\widehat{\alpha_1}}_{\widehat{\alpha_2}}>\alpha \equiv P\widehat{\beta} \dagger \widehat{y}([Q\rangle^{\widehat{\alpha_1}}_{\widehat{\alpha_2}}>\alpha)$ with $\alpha_1, \alpha_2 \in fn(Q)$

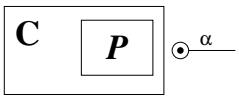
---

weak-general

(1)



(2)



$wg1$: $x \odot (C\{P\}) \equiv C\{x \odot P\}$ $x \notin fn(C\{P\})$

$wg2$: $(C\{P\}) \odot \alpha \equiv C\{P \odot \alpha\}$ $\alpha \notin fn(C\{P\})$

The relation $\equiv$ induces *congruence classes* on terms. We use $cl(P)$ to denote the congruence class of a term P with respect to the relation $\equiv$. Notice that each congruence class has finitely many terms, and thus finitely many possibilities to select a representative of a class.

Congruence rules satisfy some standard properties such as preservation of free names (interface preservation, as in Lafont's interaction nets [10]) and preservation of types.

It has been argued in [12] that two sequent proofs induce the same proof net if and only if one can be obtained from the other by a sequence of transpositions of independent rules, without giving details about these operations. We proceed further as we have presented *explicitly* how this transposition is done at the static level. This detailed study, presented here in the framework of ${}^*\mathcal{X}$ calculus, will enable the search for the essence of computational aspect as well, by considering reduction as reducing modulo congruence relation.

In what follows we show that the congruence rules allow us to perform restructuring of ${}^*\mathcal{X}$-terms (i.e. sequent proofs), so that the cut-names (names bound by a cut operation) are brought to the top level. Take for example an arbitrary ${}^*\mathcal{X}$-term of the form:

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

The name α might not be directly accessible (that is, α is maybe not principal name for P). Furthermore, this might hold for both names involved in the cut, α and x, which are possibly deep inside the structure of their corresponding terms. We prove that it is possible to transform the above term using congruence rules defined in the previous section[6] - to:

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

where C is a context, and α and x are principal names of $P^{\alpha} \preccurlyeq P$ and $Q^{x} \preccurlyeq Q$, respectively. In other words we can always pick at least one representative of a congruence class $cl(P\widehat{\alpha} \dagger \widehat{x}Q)$, which allows us to continue the computation at the top level.

In what follows we first formulate two lemmas which focus on a single cut-name (either x or α) at a time, followed by the main theorem which addresses both cut-names. In proving the results we will use the transitivity of $\preccurlyeq$ relation: If $P \preccurlyeq Q$ and $Q \preccurlyeq R$ then $P \preccurlyeq R$.

Lemma 1 (Left-restructuring). *For every term of the form $P\widehat{\alpha} \dagger \widehat{x}Q$, there exists a context C and a term P^{α}, where α is a principal name for P^{α} and $P^{\alpha} \preccurlyeq P$, such that*

$$P\widehat{\alpha} \dagger \widehat{x}Q \equiv C\{P^{\alpha}\widehat{\alpha} \dagger \widehat{x}Q\}$$

Proof. By induction on the structure of P and case analysis.

- The base case is when α is a principal name for P. Then we have $P^{\alpha} = P$ and $C = \{\ \}$.
- Assume that the property holds for the immediate subterms of P. The possible cases for P are

1.	$P = \widehat{y}\, M\, \widehat{\beta} \cdot \gamma,$	$\gamma \neq \alpha$
2.	$P = M\, \widehat{\beta}\, [y]\, \widehat{z}\, N$	
3.	$P = M\widehat{\beta} \dagger \widehat{y}N$	
4.&5.	$P = x{<}^{\widehat{x_1}}_{\widehat{x_2}}\langle M]$ and $P = [M\rangle^{\widehat{\beta_1}}_{\widehat{\beta_2}}{>}\beta,$	$\beta \neq \alpha$
6.&7.	$P = x \odot M$ and $P = M \odot \alpha,$	$\beta \neq \alpha$

We analyze the first 3 cases[7]:

[6] For the cases we prove here the set of congruence rules presented is sufficient. Presenting a complete proof would require to use all congruence rules.

[7] Due to a limited space.

1. $(\widehat{y}\, M\, \widehat{\beta}\cdot\gamma)\widehat{\alpha} \dagger \widehat{x}Q \overset{ec1}{\equiv} \widehat{y}\,(M\widehat{\alpha} \dagger \widehat{x}Q)\,\widehat{\beta}\cdot\gamma$
$\overset{i.h.}{\equiv} \widehat{y}\,(C'\{M^{\alpha}\widehat{\alpha} \dagger \widehat{x}Q\})\,\widehat{\beta}\cdot\gamma$
$\triangleq C\{M^{\alpha}\widehat{\alpha} \dagger \widehat{x}Q\},$
with $C\{\ \} = \widehat{y}\,(C'\{\ \})\,\widehat{\beta}\cdot\gamma$
2. (a) Case $\alpha \in M$.
$(M\,\widehat{\beta}\,[y]\,\widehat{z}\,N)\widehat{\alpha} \dagger \widehat{x}Q \overset{ci1}{\equiv} (M\widehat{\alpha} \dagger \widehat{x}Q)\,\widehat{\beta}\,[y]\,\widehat{z}\,N \overset{i.h.}{\equiv} (C'\{M^{\alpha}\widehat{\alpha} \dagger \widehat{x}Q\})\,\widehat{\beta}\,[y]\,\widehat{z}\,N$
$\triangleq C\{M^{\alpha}\widehat{\alpha} \dagger \widehat{x}Q\},$ with $C\{\ \} = (C'\{\ \})\,\widehat{\beta}\,[y]\,\widehat{z}\,N$
(b) Case $\alpha \in N$.
$(M\,\widehat{\beta}\,[y]\,\widehat{z}\,N)\widehat{\alpha} \dagger \widehat{x}Q \overset{ci2}{\equiv} (M\widehat{\alpha} \dagger \widehat{x}Q)\,\widehat{\beta}\,[y]\,\widehat{z}\,N \overset{i.h.}{\equiv} (C'\{M^{\alpha}\widehat{\alpha} \dagger \widehat{x}Q\})\,\widehat{\beta}\,[y]\,\widehat{z}\,N$
$\triangleq C\{M^{\alpha}\widehat{\alpha} \dagger \widehat{x}Q\},$ with $C\{\ \} = M\,\widehat{\beta}\,[y]\,\widehat{z}\,(C'\{\ \})$
3. (a) Case $\alpha \in M$.
$(M\widehat{\beta} \dagger \widehat{y}N)\widehat{\alpha} \dagger \widehat{x}Q \overset{cc1}{\equiv} (M\widehat{\alpha} \dagger \widehat{x}Q)\widehat{\beta} \dagger \widehat{y}N$
$\overset{i.h.}{\equiv} (C'\{M^{\alpha}\widehat{\alpha} \dagger \widehat{x}Q\})\widehat{\beta} \dagger \widehat{y}N$
$\triangleq C\{M^{\alpha}\widehat{\alpha} \dagger \widehat{x}Q\},$
with $C\{\ \} = (C'\{\ \})\widehat{\beta} \dagger \widehat{y}N$
(b) Case $\alpha \in N$.
$(M\widehat{\beta} \dagger \widehat{y}N)\widehat{\alpha} \dagger \widehat{x}Q \overset{cc2}{\equiv} M\widehat{\beta} \dagger \widehat{y}(N\widehat{\alpha} \dagger \widehat{x}Q)$
$\overset{i.h.}{\equiv} M\widehat{\beta} \dagger \widehat{y}(C'\{N^{\alpha}\widehat{\alpha} \dagger \widehat{x}Q\})$
$\triangleq C\{N^{\alpha}\widehat{\alpha} \dagger \widehat{x}Q\},$
with $C\{\ \} = M\widehat{\beta} \dagger \widehat{y}(C'\{\ \})$

The proof goes similarly for other cases.

Lemma 2 (Right-restructuring). *For every term of the form $P\widehat{\alpha} \dagger \widehat{x}Q$, there exists a context C and a term Q^{x}, whose principal name is x and $Q^{x} \preccurlyeq Q$, such that*

$$P\widehat{\alpha} \dagger \widehat{x}Q \equiv C\{P\widehat{\alpha} \dagger \widehat{x}Q^{x}\}$$

Proof. Similarly as previous lemma, by induction on the structure of Q and case analysis.

The following theorem confirms that the top level cut evaluation is complete, in a sense that there always exists a representative of a congruence class which enables cut evaluation at the top level relative to its cut-names.

It also shows that in the presence of congruence rules we do not need propagation rules. Propagation rules in the ${}^{*}\mathcal{X}$ calculus can be seen as means to perform restructuring of terms. However their existence blurs the core part of classical computation, and at the same time don't enable restructuring of normal forms (terms that do not contain cuts).

Theorem 1. *For every term of the form $P\widehat{\alpha} \dagger \widehat{x}Q$ there exists a context C and terms P^{α} and Q^{x} whose principal names are α, x respectively, and $P^{\alpha} \preccurlyeq P$,*

$Q^x \preccurlyeq Q$, *such that*

$$P\widehat{\alpha} \dagger \widehat{x}Q \equiv C\{P^{\alpha}\widehat{\alpha} \dagger \widehat{x}Q^x\}$$

Proof. By using the two previous lemmas, we construct the proof as follows:

$$\begin{aligned} P\widehat{\alpha} \dagger \widehat{x}Q &\overset{Lem.\ 1}{\equiv} C_1\{P^{\alpha}\widehat{\alpha} \dagger \widehat{x}Q\} \\ &\overset{Lem.\ 2}{\equiv} C_1\{C_2\{P^{\alpha}\widehat{\alpha} \dagger \widehat{x}Q^x\}\} \\ &\triangleq \quad C'\{P^{\alpha}\widehat{\alpha} \dagger \widehat{x}Q^x\}, \quad \text{with} \quad C'\{\ \} = C_1\{C_2\{\ \}\} \end{aligned}$$

And thus we are done.

6 Basic properties of $\equiv$

The congruence relation describes the way to perform restructuring of terms and thus an important property is preservation of free names. Second important property is type preservation, which ensures that term-restructuring defined by $\equiv$ can be seen as proof-transformation.

Theorem 2 (Free names preservation). *If P, Q are ${}^*\mathcal{X}$-terms, then the following holds: If $P \equiv Q$ then $fn(P) = fn(Q)$*

Proof. The property can be checked by treating carefully each rule.

Theorem 3 (Type preservation). *Let S be an ${}^*\mathcal{X}$-term and Γ, Δ contexts. Then the following holds: If $S :: \cdot\, \Gamma \vdash \Delta$ and $S \equiv S'$, then $S' :: \cdot\, \Gamma \vdash \Delta$*

Proof. The proof goes by checking that the property holds for equations inducing $\equiv$. We first write the typing derivation for the term on the left-hand side, and then for the term on the right-hand side of the equation. We prove several non-trivial cases.

- Observe the first rule of exporter-importer group of rules:
$\widehat{x}\,(P\,\widehat{\gamma}\ [z]\ \widehat{y}\,Q)\,\widehat{\beta}\cdot\alpha \overset{ei1}{\equiv} (\widehat{x}\,P\,\widehat{\beta}\cdot\alpha)\,\widehat{\gamma}\ [z]\ \widehat{y}\,Q$, with $x, \beta \in fn(P)$, $x \neq z$. On the one hand we have:

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

On the other hand,

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

The proof goes similarly for the other cases.

7 Conclusion and future work

We presented a detailed account of a possible step forward, towards the unveiling of the essential content of sequent classical proofs. This is done by declaring which syntactically different terms should be considered the same, i.e., by means of congruence rules which induce a congruence relation on terms.

The framework for this is classical logic formalized in a standard sequent system with explicit structural rules, namely weakening and contraction. Indeed it was necessary to have all the details, structural as well as logical, to be able to state what are the unessential aspects and thus extract the essence in a systematic way.

Clearly this is a base for studying the operational semantics towards defining a system with only the essential reduction rules, where reducing is reducing modulo congruence relation. The congruence relation is proved to satisfy the minimal requirement of free names preservation, as well as that of type preservation.

It is also possible to define a translation, call it $\mathcal{D}$, from terms to diagrams in a natural way, inductively on the structure of terms (basic study is given in [18]). A similar approach was taken in [9], when interpreting intuitionistic λlxr-terms (featuring explicit weakening and contraction) into proof-nets [7]. It is reasonable to expect the translation $\mathcal{D}$ to satisfy

- $P_1 \equiv P_2 \quad \text{then} \quad \mathcal{D}(P_1) = \mathcal{D}(P_2)$
- $P_1 \overset{\mathcal{D}}{\to} P_2 \quad \text{then} \quad \mathcal{D}(P_1) \overset{\mathcal{D}}{\to} \mathcal{D}(P_2)$

where P_1 and P_2 are ${}^{*}\mathcal{X}$-terms, and where $\overset{\mathcal{D}}{\to}$ is a suitable reduction relation on diagrams. However this should be carefully studied and belongs in the domain of future work.

Moreover, in terms of future work, this computational model may be relevant in relation to process session types and concurrency, as recent studies have shown that both linear intuitionistic [3] and linear classical logic [16], have a natural interpretation as protocols for concurrent process communication.

Acknowledgments

This paper was made possible by NPRP 7-988-1-178 from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the authors.

References

1. Barbanera, F., Berardi, S.: *A Symmetric Lambda Calculus for "Classical" Program Extraction*, TACS, pp. 495-515, 1994.
2. van Bakel, S., Lengrand, S. and Lescanne, P.: *The language $\mathcal{X}$: circuits, computation and Classical Logic*, Proc. 9th Italian Conf. on Theoretical Computer Science, vol. 3701, pp. 81-96, 2005.
3. Caires, L., Pfenning, F.: *Session Types as Intuitionistic Linear Propositions*, CONCOUR 2010, Lecture Notes in Computer Science, **6269** (2010), pp. 222–236.
4. Curien P.-L., Herbelin, H.: *The duality of computation*. In: Proc. 5 th ACM SIGPLAN Int. Conf. on Functional Programming (ICFP'00). ACM, 2000, pp. 233–243.
5. Gentzen, G., *Untersuchungen über das logische* Schließen. Math. **39**, (1935) 176–210.
6. Ghilezan, S., Lescanne, P, and Žunić, D.: *Computational interpretation of classical logic with explicit structural rules*, draft.
7. Girard, J.-Y.: *A new constructive logic: classical logic*, Mathematical Structures in Computer Science, **1-3** (1991), pp. 255–296.
8. Herbelin, H.: *C'est maintenant qu'on calcule, au cœur de la dualité*, Université Paris XI (Habilitation à diriger les recherches), 2005.
9. Kesner, D., Lengrand, S.: *Resource operators for lambda-calculus*, Information and Computation, vol. 205, pp. 419-473, 2007.
10. Lafont, Y.: *Interaction nets*. Proceedings of the 17th ACM symposium on Principles of programming languages, POPL, ACM Press, pp. 99–108, 1990.
11. Lengrand, S.: *Call-by-value, call-by-name, and strong normalization for the classical sequent calculus*, Electronic Notes in Theoretical Computer Science, **86-4** (2003), pp. 714–730.
12. Robinson, E.: *Proof Nets for Classical Logic*, Journal of Logic and Computation, **13-5** (2003), pp. 777–797.
13. Troelstra, A. S., Schwichtenberg, H.: *Basic Proof Theory*, Cambridge University Press, 1996.
14. Urban, C.: *Classical Logic and Computation*, University of Cambridge (Ph.D. thesis), 2000.
15. Urban, C., Bierman, G. M.: *Strong normalisation of cut-elimination in classical logic*. Fundamenta Informaticae **45 (1-2)** (2001), 123–155, (appeared also at *TLCA* in 1999).
16. Wadler, P.: *Propositions as Sessions*, Proceedings of the 17th ACM SIGPLAN international conference on Functional programming, ACM SIGPLAN Notices, **47-9** (2012), pp. 273–286.
17. Whitehead, A. N., Russell, B.: *Principia Mathematica 2nd Ed.*, Cambridge University Press, 1925.
18. Žunić, D.: *Computing With Sequent and Diagrams in Classical Logic - Calculi $^*\mathcal{X}$, $©\mathcal{X}$ and $d\mathcal{X}$*, Ph.D. thesis, ENS Lyon, France, 2007. URL: `https://tel.archives-ouvertes.fr/tel-00265549`