



Proof Nets and Explicit Substitutions

Roberto Di Cosmo, Delia Kesner, Emmanuel Polonovski

► To cite this version:

Roberto Di Cosmo, Delia Kesner, Emmanuel Polonovski. Proof Nets and Explicit Substitutions. FOSSACS, 2000, Berlin, Germany. pp.63-81. hal-00384955

HAL Id: hal-00384955

<https://hal.science/hal-00384955>

Submitted on 17 May 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Proof Nets and Explicit Substitutions

Roberto Di Cosmo* Delia Kesner § Emmanuel Polonovski*

January 13, 2000

Abstract

We refine the simulation technique introduced in [10] to show strong normalization of λ -calculi with explicit substitutions via termination of cut elimination in proof nets [12]. We first propose a notion of equivalence relation for proof nets that extends the one in [9], and we show that cut elimination modulo this equivalence relation is terminating. We then show strong normalization of the typed version of the λ_l -calculus with de Bruijn indices (a calculus with full composition defined in [8]) using a translation from typed λ_l to proof nets. Finally, we propose a version of typed λ_l with named variables which helps to better understand the complex mechanism of the explicit weakening notation introduced in the λ_l -calculus with de Bruijn indices [8].

1 Introduction

This paper uses linear logic's proof nets, equipped with an extended notion of reduction, to provide several new results in the field of explicit substitutions. It is also an important step forward in clarifying the connection between explicit substitutions and proof nets, two well established formalisms that have been used to gain a better understanding of the λ -calculus over the past decade. On one side, explicit substitutions provide an intermediate formalism that - by decomposing the β rule into more atomic steps - allows a better understanding of the execution models. On the other side, linear logic decomposes the intuitionistic logical connectives, like the arrow, into more atomic, resource-aware connectives, like the linear arrow and the explicit erasure and duplication operators given by the exponentials: this decomposition is reflected in proof nets, which are the computational side of linear logic, and provides a more refined computational model than the one given by the λ -calculus, which is the computational side of intuitionistic logic¹.

The pioneer calculus with explicit substitutions, λ_σ , was introduced in [1] as a bridge between the classical λ -calculus and concrete implementations of functional programming languages. An important property of calculi with explicit substitutions is nowadays known as PSN, which stands for “Preservation of Strong Normalization”: a calculus with explicit substitutions has PSN when all λ -terms that are strongly normalizing using the traditional β -reduction rule are also strongly normalizing w.r.t. the more refined reduction system defined using explicit substitutions. But λ_σ does *not* preserve β -strong normalization as shown by Mellies, who exhibited a well-typed term which, due to the substitution composition rules in λ_σ , is not λ_σ -strongly normalizing [17].

*DMI-LIENS (CNRS URA 1347) Ecole Normale Supérieure - 45, Rue d'Ulm - 75230 Paris France.
Email:{dicosmo,polonovs}@ens.fr

§LRI (CNRS URA 410) - Bât 490, Université de Paris-Sud - 91405 Orsay Cedex, France.
Email:kesner@lri.fr

¹Using various translations of the λ -calculus into proof nets, new abstract machines have been proposed, exploiting the Geometry of Interaction and the Dynamic Algebras [13, 2, 5], leading to the works on optimal reduction [14, 16].

Since then, a quest was started to find an “optimal” calculus having all of a wide range of desired properties: it should preserve strong normalization, but also be confluent (in a very large sense that implies the ability to compose substitutions), and its typed version should be strongly normalizing.

Meanwhile, in the linear logic community, many studies focused of the connection between λ -calculus (without explicit substitutions) and proof nets, trying to find the proper variant or extension of proof nets that could be used to cleanly simulate β -reduction, like in [7].

Finally, in [10], the first two authors of this work showed for the first time that explicit substitutions could be tightly related to linear logic’s proof nets, by providing a translation into a variant of proof nets from λ_x [18, 4], a simple calculus with explicit substitutions and named variables, but no composition.

This connection was promising because proof nets seem to have many of the properties which are required of a “good” calculus of explicit substitutions, and especially the strong normalization in the presence of a reduction rule which is reminiscent of the composition rule at the heart of Mellies’ counterexample. But [10] only dealt with a calculus without composition, and the translation was complex and obscure enough to make the task of extending it to the case of a calculus with composition quite a daunting one.

In this paper, we can finally present a notion of reduction for Girard’s proof nets which is flexible enough to allow a natural and simple translation from David and Guillaume’s λ_l , a complex calculus of explicit substitution with de Bruijn indices and full composition [8]. This translation allows us to prove that typed λ_l is strongly normalizing, which is a new result confirming a conjecture in [8]. Also, the fact that in the translation all information about variable order is lost suggests a version of typed λ_l with named variables which is immediately proved to be strongly normalizing. This is due to the fact that only the type information is used in the translation of both calculi. Also, the typed named version of λ_l gives a better understanding of the mechanisms of labels existing in the calculus. In particular, names allow to understand the fine manipulation of explicit weakenings in λ_l without entering into the complicate details of renaming used in a de Bruijn setting.

The paper is organized as follows: we first recall the basic definitions of linear logic and proof nets and we introduce our refined reduction system for proof nets (Section 2), then prove that it is strongly normalizing (Section 3). In Section 4 we recall the definition of the λ_l calculus with its type system, present the translation into proof nets, and show strong normalization of typed λ_l . Finally, we introduce a version of typed λ_l with named variables (Section 5), enjoying the same good properties, and we conclude with some remarks and directions for future work (Section 7).

2 Linear logic, proof nets and extended reduction

We recall here some classical notions from linear logic, namely the linear sequent calculus and proof nets, and some basic results concerning confluence and normalization.

MELL: Multiplicative Exponential linear logic Let \mathcal{A} be a set of *atomic formulae*. We suppose that \mathcal{A} is partitioned in two disjoint subsets representing *positive* and *negative* atoms respectively.

The set of formulae of the Multiplicative Exponential fragment of linear logic (called MELL) is defined by the following grammar, where $a \in \mathcal{A}$:

$$\mathcal{F} ::= a \mid \mathcal{F} \otimes \mathcal{F} \text{ (tensor)} \mid \mathcal{F} \wp \mathcal{F} \text{ (par)} \mid !\mathcal{F} \text{ (of course)} \mid ?\mathcal{F} \text{ (why not)}$$

For every $p \in \mathcal{A}$, we assume that there is $p' \in \mathcal{A}$, called the *linear negation of the atom* p . Linear negation of formulae is *defined* as follows

$$p^\perp = p' \quad p'^\perp = p \quad A^{\perp\perp} = A \quad (?A)^\perp = !(A^\perp) \quad (A \otimes B)^\perp = A^\perp \wp B^\perp$$

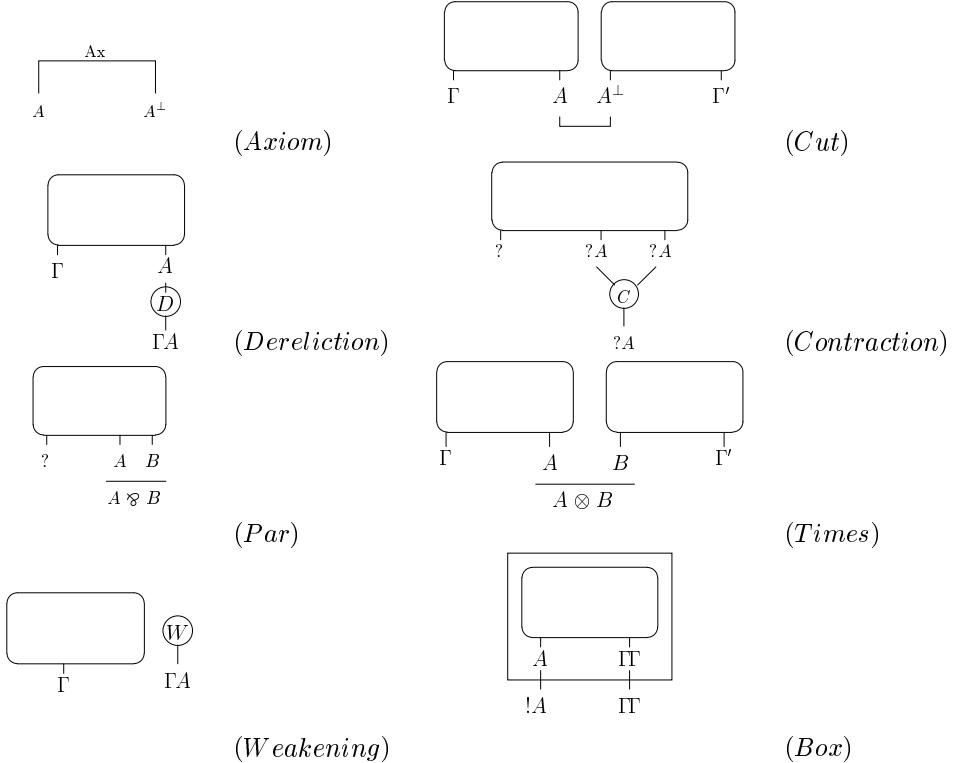
From now on we will write p^\perp instead of p' .

The name MELL comes from the connectors \otimes and \wp which are called “multiplicatives”, while $!$ and $?$ are called “exponentials”. We say that a formula is exponential if it starts with an exponential connector. While we refer the interested reader to [12] for more details on linear logic, we give here a one-sided presentation of the sequent calculus for MELL:

$$\begin{array}{c} \frac{}{\vdash A, A^\perp} \text{Axiom} \quad \frac{\vdash ?, A \quad \vdash A^\perp, \Delta}{\vdash ?, \Delta} \text{Cut} \quad \frac{\vdash ?, A}{\vdash ?, ?A} \text{Dereliction} \quad \frac{\vdash ?, ?A, ?A}{\vdash ?, ?A} \text{Contraction} \\[10pt] \frac{\vdash ?, A, B}{\vdash ?, A \wp B} \text{Par} \quad \frac{\vdash ?, A \quad \vdash B, ?'}{\vdash ?, A \otimes B, ?'} \text{Times} \quad \frac{\vdash ?}{\vdash ?, ?A} \text{Weakening} \quad \frac{\vdash A, ??}{\vdash !A, ??} \text{Box} \end{array}$$

MELL proof nets To all sequent derivations in MELL it is possible to associate an object called a “proof net”, which allows to abstract from many inessential details in a derivation, like the order of application of independent logical rules: for example, there are many inessentially different ways to obtain $\vdash A_1 \wp A_2, \dots, A_{n-1} \wp A_n$ from $\vdash A_1, \dots, A_n$, while there is only one proof net representing all these derivations.

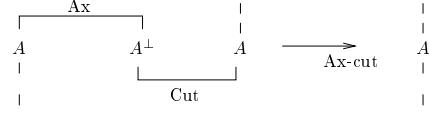
Proof nets are defined inductively by rules that follow closely the ones of the one-sided sequent calculus, and the set of proof nets is denoted PN . To simplify the drawing of a proof net, we use the following notation: a conclusion with a capital greek letter $?, \Delta, \dots$ really stands for a set of conclusions, each one with its own wire.



Each box has exactly one conclusion preceded by a $!$, which is named “principal” port (or formula), while the other conclusions are named “auxiliary” ports (or formulae). In what follows, we will sometimes write an axiom link as $A \longrightarrow A^\perp$.

Reduction of proof nets Proof nets are the “computational object” behind linear logic, because there is a notion of reduction on them (called also “cut elimination”) that corresponds to the cut-elimination procedure on sequent derivations. The traditional reduction system for MELL is defined as follows:

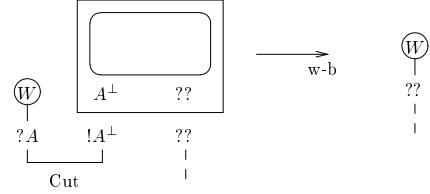
Reduction acting on a cut $Ax - cut$, removing an axiom :



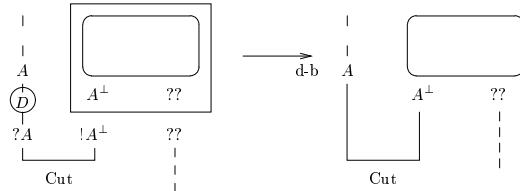
Reduction acting on a cut $\wp - \otimes$:



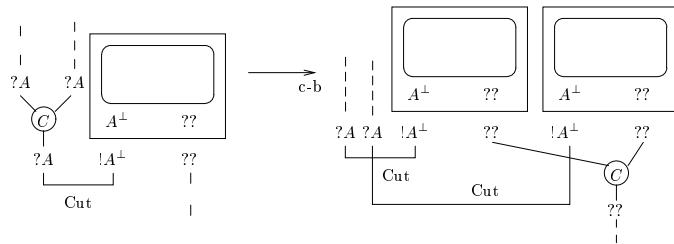
Reduction acting on a cut $w - b$, erasing a box :



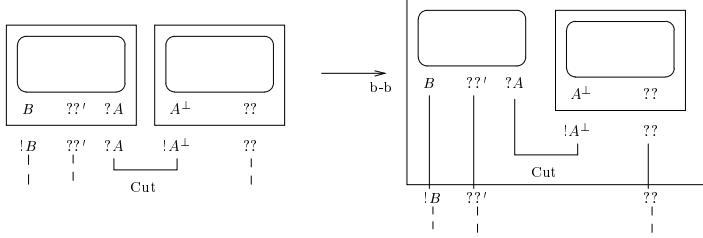
Reduction acting on a cut $d - b$, opening a box :



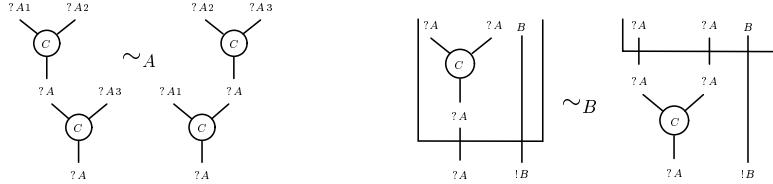
Reduction acting on a cut $c - b$, duplicating a box :



Reduction acting on a cut $b - b$, absorbing a box into another :



Extended reduction modulo an equivalence relation Unfortunately, the original notion of reduction on PN is not well adapted to simulate neither the β rule of λ -calculus, nor the rules dealing with propagation of substitution in explicit substitution calculi: too many inessential details on the order of application of the rules are still present, and to make abstraction from them, one is naturally led to define an equivalence relation on PN , as is done in [9], where the following two equivalences are introduced:



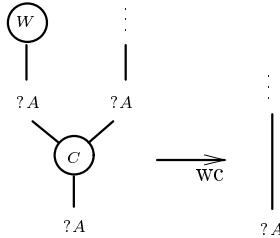
Equivalence A turns contraction into an associative operator, and corresponds to forgetting the order in which the contraction rule is used to build, for example, the derivation:

$$\frac{\vdash ?A, ?A, ?A}{\frac{\vdash ?A, ?A}{\vdash ?A}} \text{ Contraction}$$

Equivalence B abstracts away the relative order of application of the rules of box-formation and contraction on the premises of a box, like in the following example.

$$\frac{\begin{array}{c} \vdash ?A, ?A, B \\ \vdash ?A, B \end{array}}{\vdash ?A, !B} \text{ Box} \quad \frac{\vdash ?A, ?A, B}{\frac{\vdash ?A, ?A, !B}{\vdash ?A, !B}} \text{ Box}$$

Finally, besides the equivalence relation defined in [9], we will also need an extra reduction rule allowing to remove unneeded weakening links when simulating explicit substitutions:



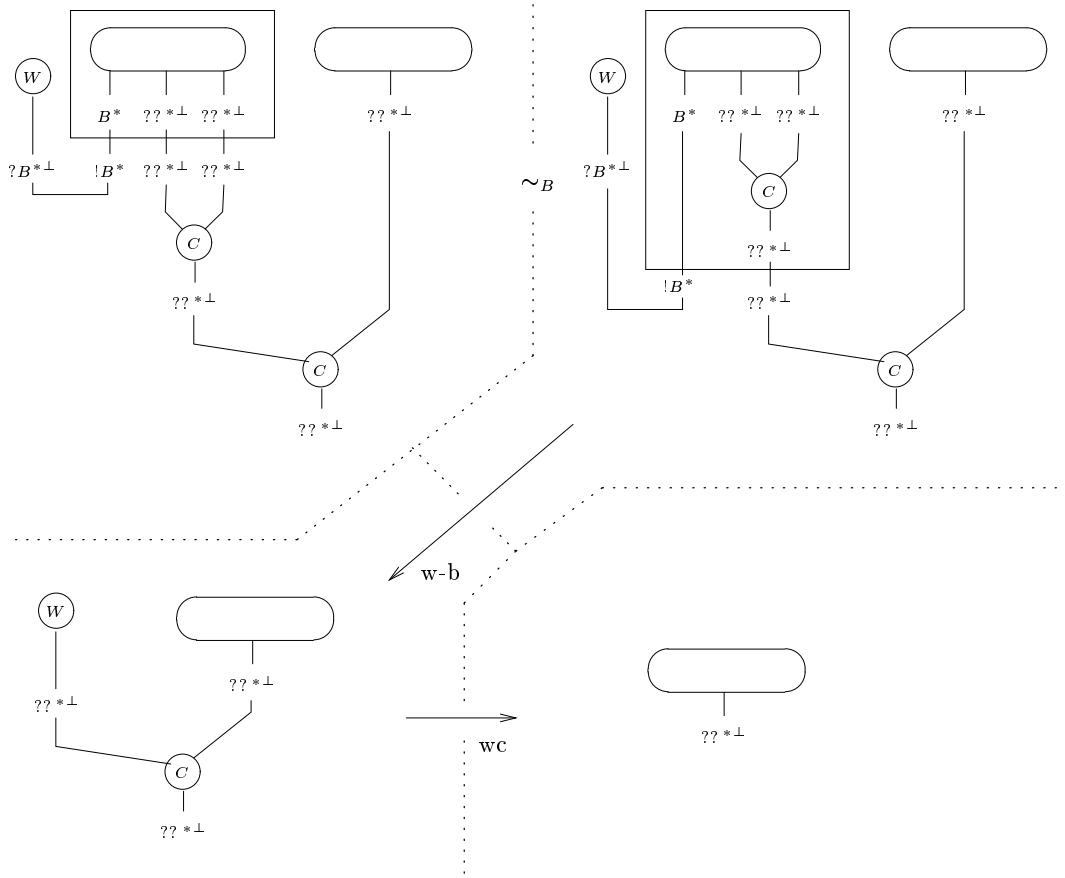
This rule allows to simplify the proof below on the left into the proof on the right

$$\frac{\frac{\pi}{\vdash ?A}}{\vdash ?A, ?A} \text{ Weakening} \quad \frac{\vdash ?A}{\vdash ?A} \text{ Contraction} \quad \frac{\pi}{\vdash ?A}$$

Notation We will call in the following R the system made of rules $Ax-cut$, $\wp - \otimes$, $w - b$, $d - b, c - b$, $b - b$ and wc ; we will name E the relation induced on PN by the contextual closure of axioms A and B ; we will write R_E for the system made of the rules in R and the equivalences in E ; finally, R_E^{-wc} will stand for system R_E without rule wc .

Systems R_E and R_E^{-wc} , that contain E , are actually defining a notion of *reduction modulo an equivalence relation*, so we write for example $t \rightarrow_{R_E} s$ if and only if there exist r' and s' such that $r =_E r' \rightarrow_R s' =_E s$, where the equality $=_E$ is the reflexive, symmetric and transitive closure of the relation defined by A and B .

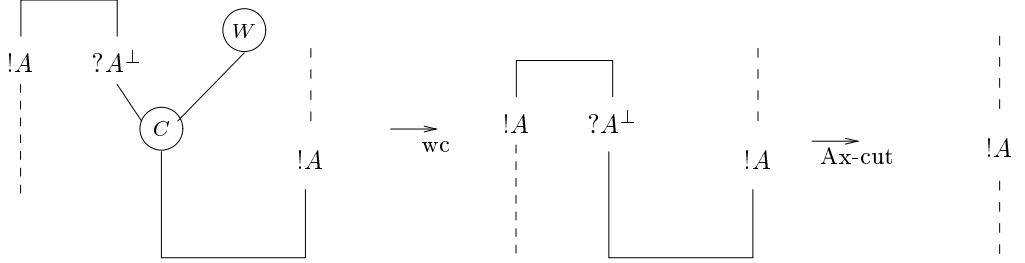
An example of reduction in R_E is given here:



The reduction R_E is flexible enough to allow an elegant simulation of β reduction and of explicit substitutions, but for that, we first need to establish that R_E is strongly normalizing. Let us see this property in the next section.

3 Termination of R_E

We know from [9] that R_E^{-wc} is terminating, and we can show easily that wc is terminating too, so if we could show that the wc -rule can be postponed with respect to all the other rules of R_E^{-wc} , we would be easily done using a well-known abstract lemma. Unfortunately, there is precisely one case in which we cannot postpone the wc -rule: when a wc reduction creates an axiom-cut redex, which in turn can only happen if the axiom link in question introduces an exponential formula as the following example of reduction sequence from t to t' shows:



We remark that starting from the same proof net t we cannot get the same result t' by delaying the application of the wc -rule. This is due to the fact that the axiom involved in t has exponential formulae and this is the only case in which the wc -rule cannot be postponed. As a consequence we can split our termination proof in the following way:

- We first prove that R_E is terminating on the set of proof nets *without exponential axioms* (Theorem 3.7). This can be done with the postponement technique (Lemma 3.3), because the counterexample given above only hold in proof nets with exponential axioms.
- We then show that termination of R_E on all proof nets in PN is a consequence of termination of R_E on proof nets without exponential axioms (Theorem 3.11). To obtain this result, we give a translation of any proof net in PN into a proof net without exponential axioms which allows us to simulate the reduction notion R_E on PN into the reduction notion R_E on proof nets without exponential axioms.

3.1 Termination of R_E on proof nets without exponential axioms

We show in this section that all the R_E -reduction sequences from a proof net without exponential axioms terminate. We first remind the following result from [9]:

Lemma 3.1 (Termination of R_E^{-wc}) The relation $\rightarrow_{R_E^{-wc}}$ is terminating on PN .

Then, we establish the termination of wc .

Lemma 3.2 (Termination of wc) The relation \rightarrow_{wc} is terminating on PN .

Proof. The wc -rule strictly decreases the number of nodes in a proof net so no infinite wc -reduction sequence is possible. ▀

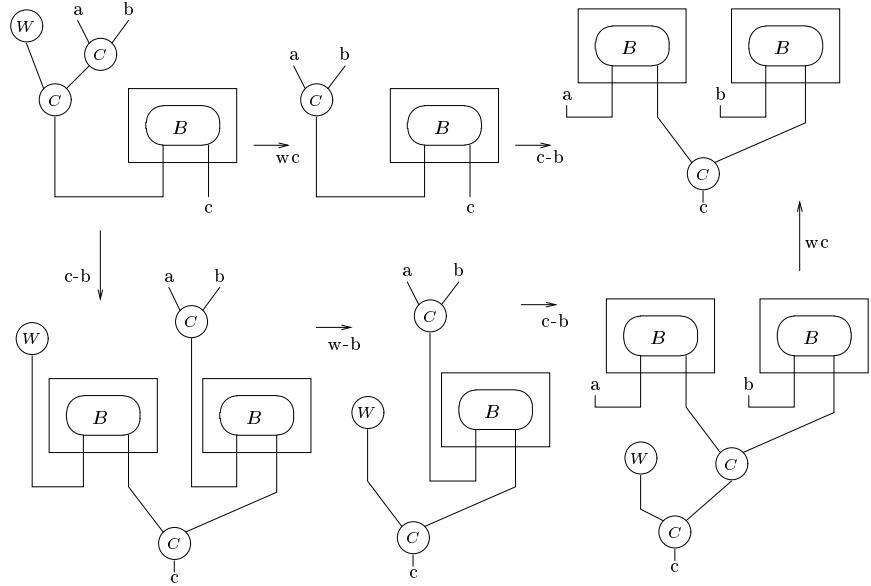
Finally, we show that given any proof net without exponential axioms, the wc -rule can be postponed with respect to any rule of R_E^{-wc} .

Lemma 3.3 (Postponement of wc w.r.t R_E^{-wc}) Let t be a proof net without exponential axioms. If $t \rightarrow_{wc} \rightarrow_{R_E^{-wc}} t'$, then, there is a sequence $t \rightarrow^{+}_{R_E^{-wc}} \rightarrow^{*}_{wc} t'$.

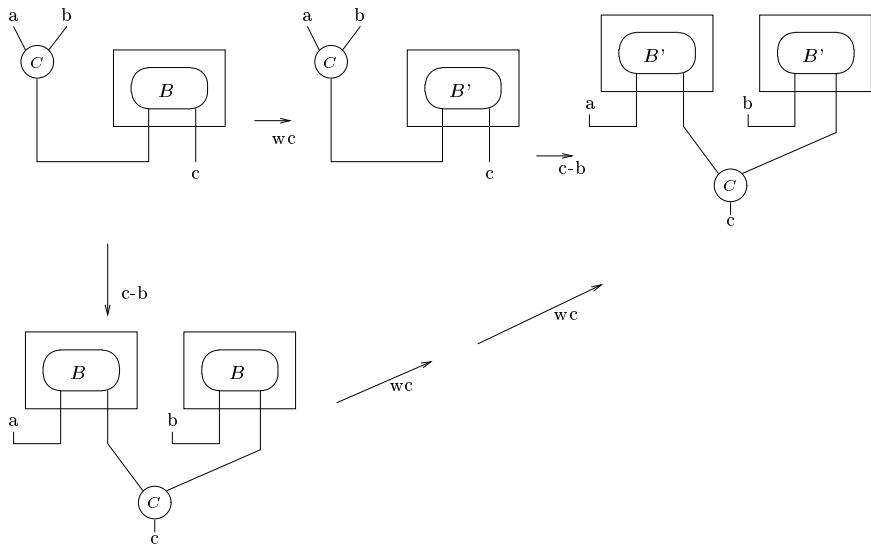
Proof. Let $t \rightarrow_{wc} \rightarrow_{R_E^{-wc}} t'$ be a reduction sequence starting at t with a wc -reduction step. Let us show that we can build an equivalent reduction $t \rightarrow^{+}_{R_E^{-wc}} \rightarrow^{*}_{wc} t'$ by analyzing all the possible cases.

We do not detail here the cases of disjoint redexes: if we apply the wc -rule followed by a rule $R1$ in R_E^{-wc} and if the redexes occur at disjoint positions, then it is evident that $R1$ can be applied first, followed by wc , and getting the same result. We study now all the remaining cases:

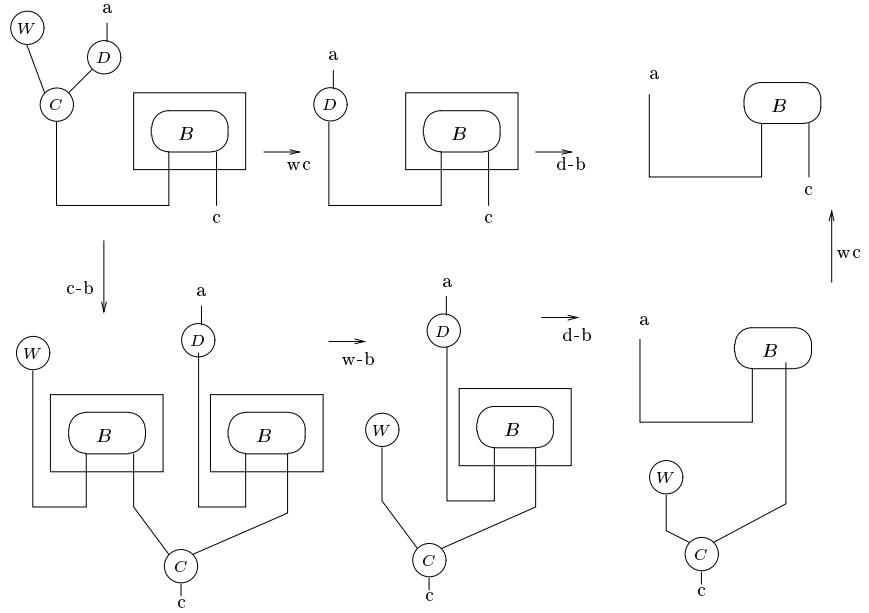
1. The rule $c - b$, first possibility :



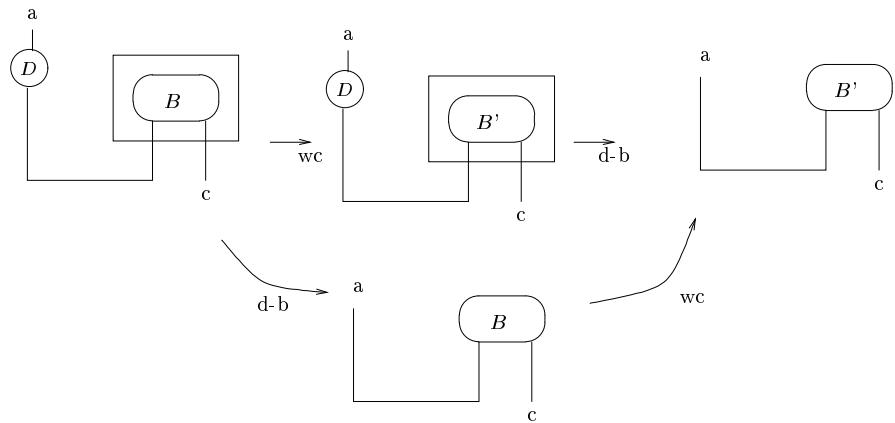
2. The rule $c - b$, second possibility :



3. The rule $d - b$, first possibility :

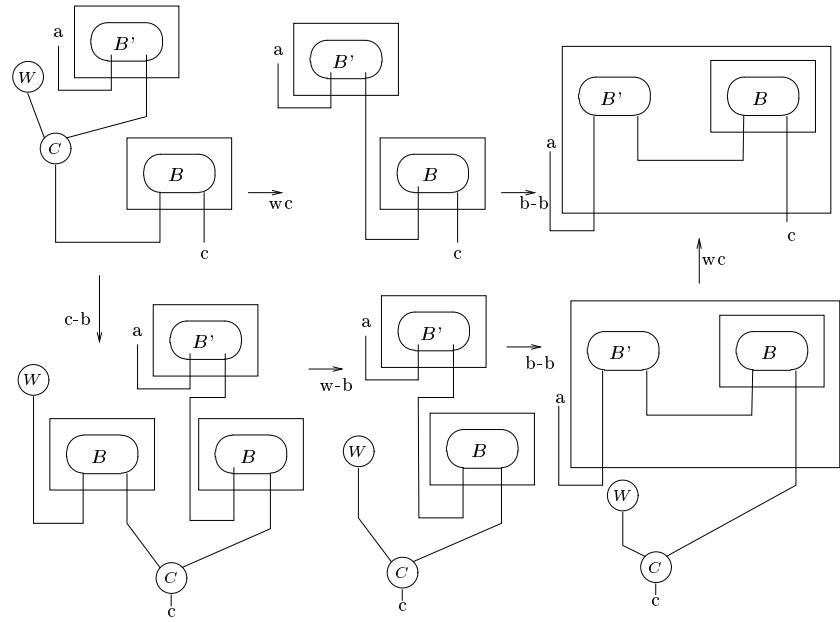


4. The rule $d - b$, second possibility :



We notice that everything is happening as if the redexes were disjoint. This is due to the fact that the $d - b$ rule is non-duplicating and non-erasing w.r.t boxes. As a consequence, the wc -redex is still preserved after the application of the $d - b$ rule.

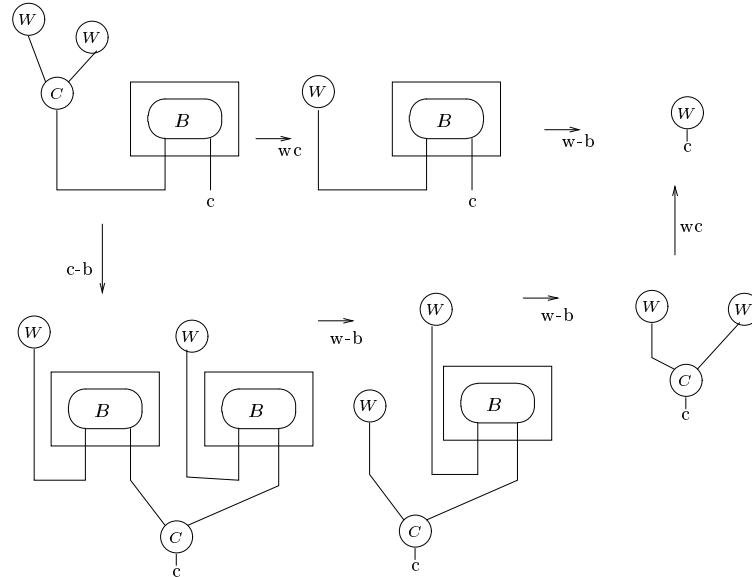
5. The rule $b - b$, first possibility :



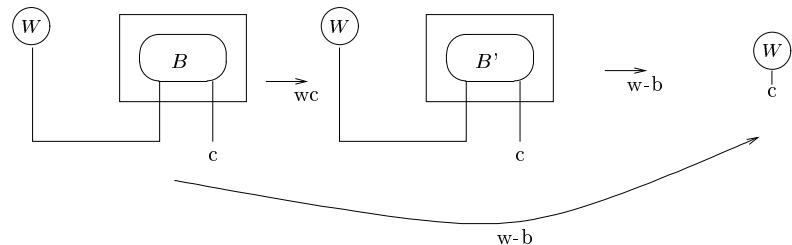
6. The rule $b - b$, second possibility :

For the same reason as for $d - b$, the redexes are considered as disjoint.

7. The rule $w - b$, first possibility :



8. The rule $w - b$, second possibility :



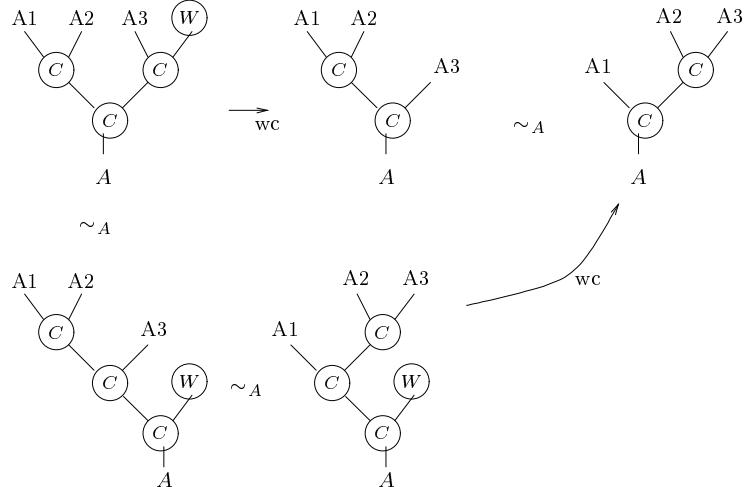
9. The other rules :

For all the other rules the redexes are disjoint. Indeed, only the contraction or the weakening appearing in a reduction sequence can interact with another redex. However, during the elimination of the axiom-cut or the $\otimes\otimes$ cut, neither the weakening nor the contraction may interfere (for proof nets without exponential axioms).

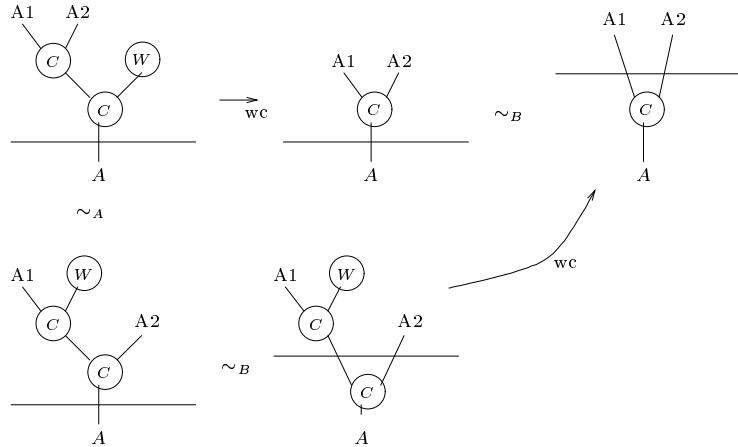
?????????

Until now we have only worked with reduction rules of R_E , but to complete our statement we also need to show that the wc -rule can be delayed w.r.t one equivalence step. We proceed as we did for the reduction rules. We do not study the cases where redexes are disjoint because they are evident. The remaining cases are the following:

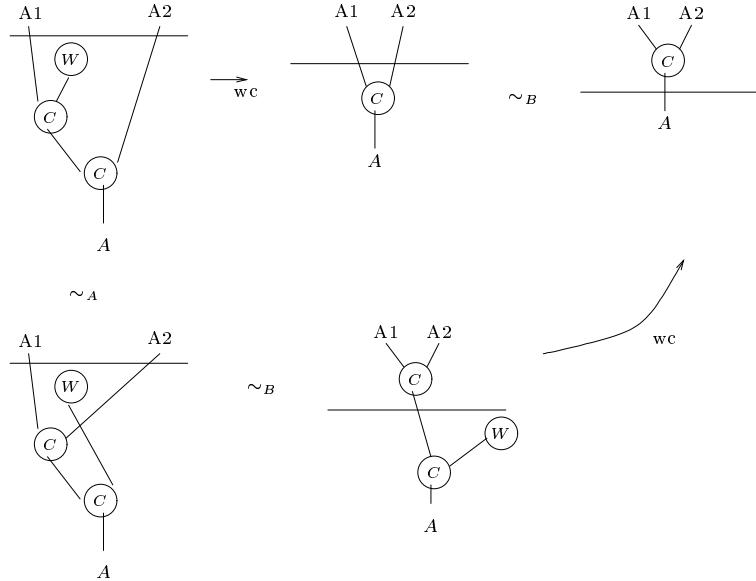
1. Associativity :



2. Box passing, first case :



3. Box passing second case :



We can now put together the previous results to prove termination of R_E on the set of proof nets without exponential axioms.

But first of all, we establish two propositions stating the stability of wc and R_E^{-wc} .

Proposition 3.4 (Stability by \rightarrow_{wc}) The set of the proof nets without exponential axioms is stable by the reduction \rightarrow_{wc} . In other words, for every proof net r without exponential axioms, if $r \rightarrow_{wc} r'$ then r' is a proof net without exponential axioms.

Proof. We know that the application of wc does not modify the formulae of a proof net. As a consequence, it is easy to see that the set of proof nets without exponential axioms is stable by \rightarrow_{wc} .

Proposition 3.5 (Stability by $\rightarrow_{R_E^{-wc}}$) The set of the proof nets without exponential axioms is stable by the reduction $\rightarrow_{R_E^{-wc}}$. In other words, for every proof net r without exponential axioms, if $r \rightarrow_{R_E^{-wc}} r'$ then r' is a proof net without exponential axioms.

Proof. The cut-elimination rules are local reductions so that they do not add axioms to the a proof net (and so, *a priori* no exponential axioms). As a consequence, the formulae of a proof net are never modified and, if r does not have exponential axioms, then r' can neither have exponential axioms.

Mettre l'endroit où on applique de la stability

Lemma 3.6 (Extraction of R_E^{-wc}) Let S be an infinite sequence of R_E -reductions starting at a proof net t without exponential axioms. Then, there is a sequence of R_E -reductions from the same proof net t which starts by $t \rightarrow_{R_E^{-wc}} t'$, where t' is also a proof net without exponential axioms, and which continues with an infinite sequence S' . We write this sequence as $(t \rightarrow_{R_E^{-wc}} t') \cdot S'$.

Proof. Let S be an infinite sequence of R_E -reductions starting at t :

$$t \longrightarrow_{R_E} \dots \longrightarrow_{R_E} \dots \longrightarrow_{R_E} \dots$$

We know, by Lemmas 3.2 and 3.1, that the systems wc and R_E^{-wc} are both terminating, so it is not possible to have an infinite sequence only made of wc or R_E^{-wc} . As a consequence, the infinite sequence of R_E -reductions must be an infinite alternation of non-empty finite sequences of wc and R_E^{-wc} .

Now, there are two cases: either the alternation of sequences starts with a sequence of R_E^{-wc} -reductions, and then the result holds by taking the sequence S without its first reduction step as S' ;

Or the alternation starts with a wc -step :

$$t \longrightarrow^+_{wc} \longrightarrow^+_{R_E^{-wc}} \longrightarrow^+_{wc} \longrightarrow^+_{R_E^{-wc}} \dots$$

that is, written in other way

$$t \longrightarrow^+_{wc} \longrightarrow_{R_E^{-wc}} t' \longrightarrow^*_{R_E^{-wc}} \longrightarrow^+_{wc} \longrightarrow^+_{R_E^{-wc}} \dots$$

In this case, we consider the sub-sequence $P = t \longrightarrow^+_{wc} \longrightarrow_{R_E^{-wc}} t'$ of the sequence S starting at t . This sub-sequence is composed by k reduction steps of wc and one reduction of R_E^{-wc} . Let call R the remaining sub-sequence of S .

By applying Lemma 3.3 k times on P , we can move the rule of R_E^{-wc} at the head of the sequence. By Propositions 3.5 and 3.4 we know that all the proof nets of this reduction sequence are proof nets without exponential axioms. We thus obtain a finite sequence P' which begins with a reduction $t \longrightarrow_{R_E^{-wc}} t'$, and ends on t'' . As a consequence, $P' \cdot R$ is the infinite sequence starting by a reduction R_E^{-wc} we were looking for. ▀

Now it is easy to establish the fundamental theorem of this section:

Theorem 3.7 (Termination of R_E on proof nets without exponential axioms) The reduction relation R_E is terminating on the set of proof nets without exponential axioms.

Proof. We show it by contradiction. Let us suppose that R_E is not terminating on those nets. Then, there exist a proof net without exponential axioms t and an infinite sequence S of R_E starting at t . By applying Lemma 3.6 to this sequence S , we obtain a sequence $(t \longrightarrow_{R_E^{-wc}} t') \cdot S'$ such that S' is infinite again. If we iterate this procedure an arbitrary number times, we obtain a sequence of R_E^{-wc} -reduction steps arbitrary long. This contradicts the fact that R_E^{-wc} is terminating. ▀

3.2 Termination of R_E on proof nets with exponential axioms

We know now that R_E is terminating on every proof net without exponential axioms, but we want now to show even more: termination of R_E on *all* the proof nets. To achieve this result, we show in this section how to associate to a proof net t , which can eventually contain some exponential axioms, another proof net $E(t)$ without exponential axioms, and such that every reduction from t of length n can be “simulated” on $E(t)$ by another reduction of length at least n . This property will be enough to reduce termination of R_E on proof nets with exponential axioms to termination of R_E on proof nets without exponential axioms.

Now, we first observe that in the case of the MLL proof nets, one can use indifferently nets with “atomic” axioms (i.e. the conclusions of the axiom links are atomic formulae) or not, because we can easily replace a non-atomic axiom by its “expansion”: for example, if A and B are two formulae, then the axiom

$$\begin{array}{c} \boxed{} \\ A \otimes B \qquad \qquad A^\perp \wp B^\perp \end{array}$$

can be replaced by

$$\begin{array}{c} \boxed{} \\ \boxed{} \\ A \qquad B \\ \hline A \otimes B \qquad \qquad A^\perp \qquad B^\perp \\ \hline A^\perp \wp B^\perp \end{array}$$

We can iterate this procedure until no non-atomic axiom remains in the proof net, and obtain in this way what we will write as $\exp(A \otimes B \quad A^\perp \wp B^\perp)$, the *complete expansion* of the axiom $A \otimes B \quad A^\perp \wp B^\perp$. An expanded proof net is still a well-formed proof net (as defined in Section 2) and has the same dynamic properties of the original proof net, i.e. all reductions on the original proof net can be simulated on the expanded proof net.

Starting from this observation, we could define expansion for MELL, and hope to obtain a similar result. Nevertheless, it turns out that for our purposes it is enough to expand *only* exponential axioms, without expanding all non atomic ones, so we will focus here on this limited expansion of exponentials.

Let us formalize this special notion of complete expansion of an axiom, for which we can associate a measure of complexity rk .

Definition 3.1 (Complete expansion of an axiom link) For each axiom link $\overline{A \quad A^\perp}$ we can associate a net $\exp(\overline{A \quad A^\perp})$ with same conclusions, defined by induction on the complexity of the formula A as follows:

- $\exp(\overline{A \quad A^\perp}) = \overline{A \quad A^\perp}$, if A is not an exponential formula

$$\bullet \exp(\overline{!A \quad ?A^\perp}) = \boxed{\begin{array}{c} \boxed{} \\ \exp(\quad A \quad A^\perp \quad) \\ | \\ !A \quad ?A^\perp \end{array}}$$

which is well defined, because the formula A is smaller than $!A$.

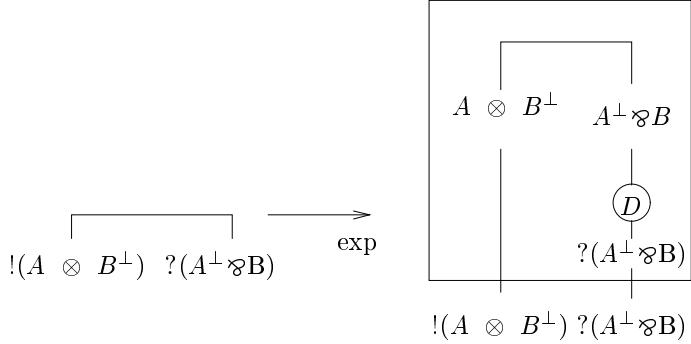
Definition 3.2 (Measure of a complete expansion) We define the measure rk of a complete expansion of an axiom by cases:

- $rk(\exp(\overline{A \quad A^\perp})) = 0$, if A is not an exponential formula
- $rk(\exp(\overline{?A^\perp \quad !A})) = 1 + rk(\exp(\overline{A \quad A^\perp}))$

We can now define the notion of expanded net $E(t)$ for every net t :

Definition 3.3 (Expanded net) The *expanded net of a net t*, written $E(t)$, is the proof net obtained from t by replacing each occurrence of an exponential axiom a by $\exp(a)$.

For example, here is a proof net and its expanded net:



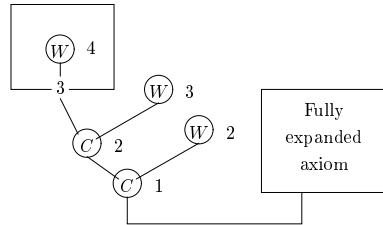
The measure of the complete expansion of this net is 1.

Remark 3.8 The only difference between a proof net t and its expanded net $E(t)$ is on the set of their axioms. So, for every reduction $t \rightarrow_{R_E} t'$ which does not affect the axioms of t , there is a reduction $E(t) \rightarrow_{R_E} E(t')$.

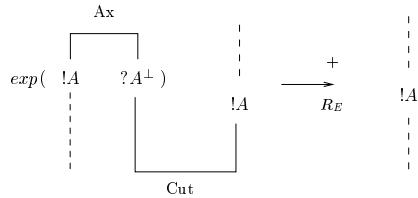
We have now to show that there is no problem for the axioms either, and to do so we need the following measure:

Definition 3.4 (Maximal distance of a cut) Given a proof net t and a cut link on a completely expanded axiom a in t , the measure $d(a, t)$ is the maximal distance, in the proof net t , between this cut and the first weakening or dereliction node encountered in the way which leaves the cut, by the opposite extremity from the expanded axiom a , and go throw the nodes from down to up (here up and down are used formally for the orientation of the nodes presented in the introduction). More precisely, each node encountered and each box passed on the way values 1, including the final dereliction or weakening node. This measure is always finite on a finite proof net because there are no arbitrary long ascendant ways.

Example 3.9 In the following net, the maximal distance of the cut is 4.



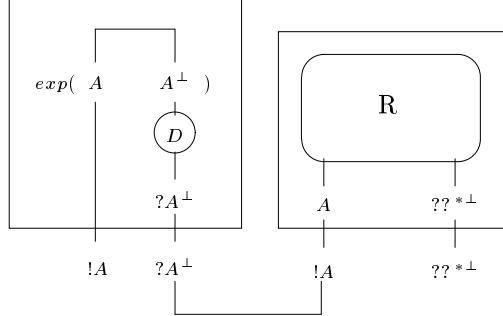
Lemma 3.10 (Cut elimination on an expanded net) Let t be an expanded net. A cut in t with a completely expanded axiom $\text{exp}(a)$ reduces in t like in an ordinary axiom cut. In other words,



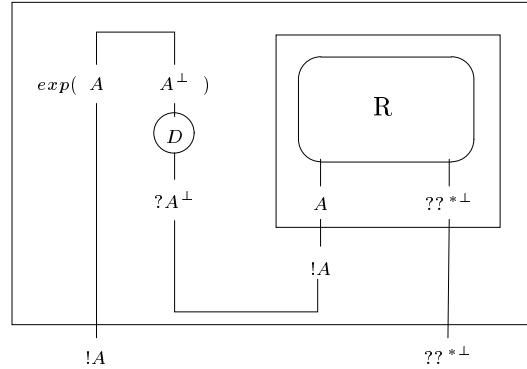
Proof. We prove the property by induction on the lexicographic order $(\text{rk}(\text{exp}(a)), d(\text{exp}(a), t))$ where $\text{exp}(a)$ is the completely expanded axiom in the proof net t .

All the cases such that $rk(exp(a)) = 0$ (including the base case) correspond to a proof net in which $exp(a)$ is an axiom link, so the same reduction rule applies and the property then trivially holds. Otherwise, for $rk(exp(a)) > 0$, we must consider all the other cases, which are separated in two different categories:

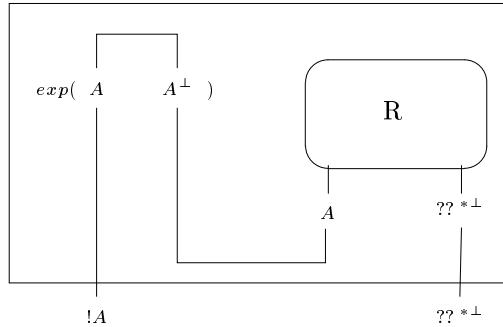
The cut formula has the following shape : $?A^\perp$. In this case, the proof net is expanded, so the cut must necessarily take place with the principal door of a box :



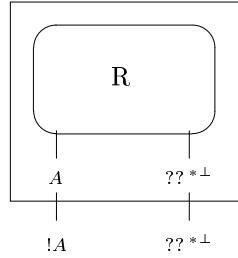
then, we can eliminate the $b - b$ cut to obtain :



then, we eliminate the $d - b$ cut :

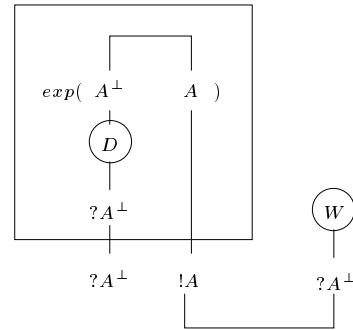


finally, by definition, $\overline{exp(A^\perp | A)}$ is smaller than $\overline{exp(?A^\perp | !A)}$. So, the measure rk has decreased and we can apply the hypothesis induction on the axiom cut connected with $\overline{exp(A^\perp | A)}$. Then we obtain :



The cut formula has the following shape : $!A$. In this case, there are several possibilities for the node which is cut with the completely expanded axiom:

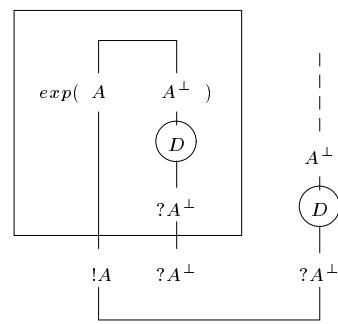
- A weakening node. In this case, it is not necessary to use the induction...



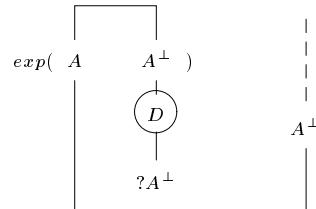
We eliminate the $w - b$ cut to obtain :



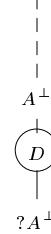
- A dereliction node. The cut has the following shape :



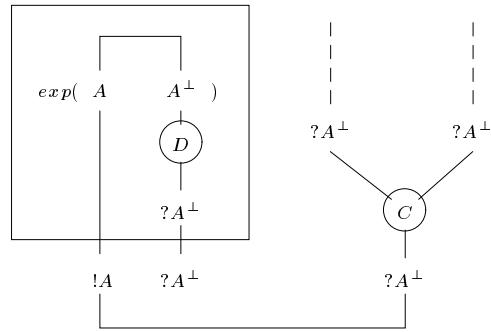
and we can eliminate the $d - b$ cut :



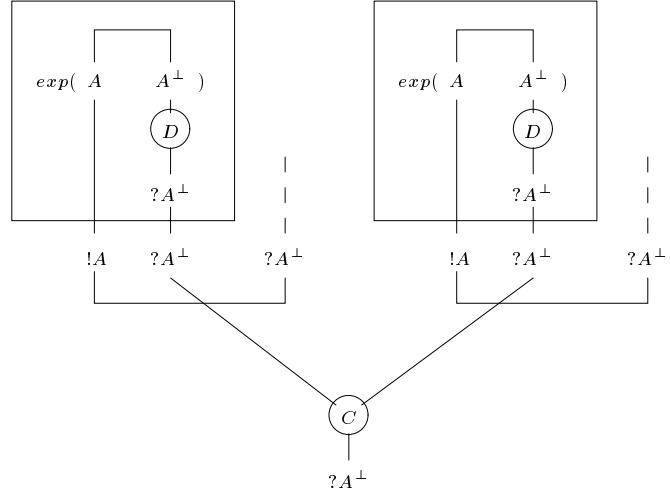
then, as previously, we notice that the measure $rk(\overline{\exp(A^\perp - A)})$ is smaller than the measure $rk(\overline{\exp(?A^\perp - !A)})$ so we can apply the induction to obtain:



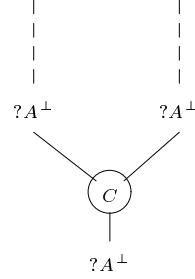
- A contraction node. In this case, the proof net has the following shape:



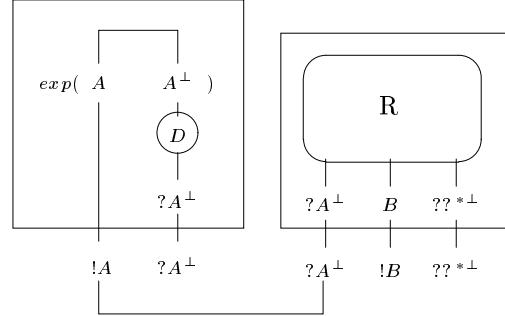
and we can then eliminate the $c - b$ cut as follows:



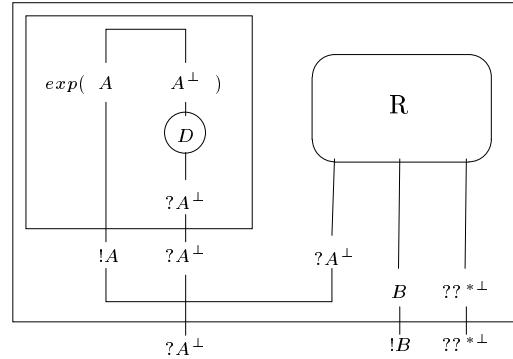
Now, the measure $rk(\overline{\exp(A^\perp - A)})$ is the same but d has decreased in the new proof net (we have one less contraction node), so that we can apply twice the induction hypothesis to obtain the final proof net:



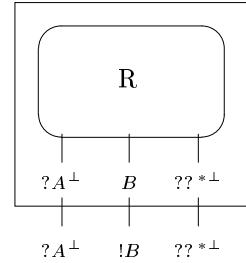
- An auxiliary door of a box. In this last case, we have this configuration :



so that we can eliminate the $b - b$ cut as follows:



Now, the measure $rk(\overline{\exp(A^\perp \quad A)})$ is the same but d has decreased in the new proof net (we have one less box), so that we can apply the induction hypothesis to obtain the final proof net:



- Since the proof net is expanded there is not cut with an exponential axiom, so there is no other case to consider.

This allows us to establish the final result of this section :

Theorem 3.11 (Termination of R_E) The reduction R_E is terminating on all proof nets.

Proof. We establish this result by proving that each reduction step $t \rightarrow_{R_E} t'$ can be simulated by at least one reduction step $E(t) \rightarrow^{+}_{R_E} E(t')$.

If the reduction step $t \rightarrow_{R_E} t'$ does not reduce any exponential axiom with a cut, then we obtain the result immediately because the only difference between t and $E(t)$ is on their axioms. Indeed, we can reproduce the same reduction on $E(t)$ in order to obtain $E(t')$ and this concludes this case.

Otherwise, if $t \rightarrow_{R_E} t'$ reduces an exponential axiom a with a cut then by Lemma 3.10 there exist a non-empty sequence of reductions starting at $E(t)$ which eliminates the complete expansion of the axiom a , and gives the proof net $E(t')$.

Now, to conclude the proof, suppose that there is a proof net t such that the reduction R_E is not terminating on t , that is, there is an infinite R_E -reduction sequence starting at t . By the previous remark we can simulate this infinite reduction sequence by another R_E -reduction sequence on expanded proof nets not containing exponential axioms. This leads to a contradiction with Theorem 3.7 so that we can conclude that R_E is terminating on the set of all proof nets. ▀

4 From λ_l with de Bruijn indices to PN

We now study the translation from typed terms of the λ_l -calculus [8] into proof nets. We start by introducing the calculus, then we give the translation of types of λ_l into formulae of linear logic, and the translation of terms of λ_l into linear logic proof nets PN . We verify that we can correctly simulate every reduction step of λ_l via the notion of reduction R_E . Finally, we use this simulation result to show strong normalization of the λ_l -calculus.

4.1 The λ_l -calculus

The λ_l -calculus is a calculus with explicit substitutions where substitutions are unary (and not multiple). The version studied in this section has variables encoded with de Bruijn indices. The terms of λ_l are given by the following grammar:

$M ::=$	\underline{n}	<i>variable</i>
	λM	<i>abstraction</i>
	$(M M)$	<i>application</i>
	$\langle k \rangle M$	<i>label</i>
	$[i/M, j]M$	<i>substitution</i>

Intuitively, the term $\langle k \rangle M$ means that the $k - 1$ first indices in M are not “free” (in the sense of free variables of calculus with indices). The term $[i/N, j]M$ means that the $i - 1$ first indices are not free in N and the $j - 1$ following indices are not free in M . Those indices are used to split the typing environment of $[i/N, j]M$ in three parts: the first (resp. second) one for free variables of M (resp. N), the third one for the free variables in M and N .

The de Bruijn indices we use start with $\underline{0}$ instead of $\underline{1}$. For example, the identity function is written as $I = \lambda \underline{0}$.

The reduction rules of λ_l are given in Figure 1 and the typing rules of λ_l are given in Figure 2, where we suppose that $|?| = i$ and $|\Delta| = j$.

We notice that for each well-typed term of the λ_l -calculus, there is only one possible typing judgment. This will simplify the proof of simulation of λ_l by easily considering the unique typing judgment of terms.

As expected the λ_l -calculus enjoys the subject reduction property [15].

(b_1)	(λMN)	\rightarrow	$[0/N, 0]M$
(b_2)	$(\langle k \rangle (\lambda M) N)$	\rightarrow	$[0/N, k]M$
(f)	$[i/N, j](\lambda M)$	\rightarrow	$\lambda[i + 1/N, j]M$
(a)	$[i/N, j](MP)$	\rightarrow	$([i/N, j]M)([i/N, j]P)$
(e_1)	$[i/N, j]\langle k \rangle M$	\rightarrow	$\langle j + k - 1 \rangle M$ if $i < k$
(e_2)	$[i/N, j]\langle k \rangle M$	\rightarrow	$\langle k \rangle [i - k/N, j]M$ if $i \geq k$
(n_1)	$[i/N, j]\underline{k}$	\rightarrow	\underline{k} if $i > k$
(n_2)	$[i/N, j]\underline{i}$	\rightarrow	$\langle i \rangle N$
(n_3)	$[i/N, j]\underline{k}$	\rightarrow	$\underline{j+k-1}$ if $i < k$
(c_1)	$[i/N, j][k/P, l]M$	\rightarrow	$[k/[i - k/N, j]P, j + l - 1]M$ if $k \leq i < k + l$
(c_2)	$[i/N, j][k/P, l]M$	\rightarrow	$[k/[i - k/N, j]P, l][i - l + 1/N, j]M$ if $i \geq k + l$
(d)	$\langle i \rangle \langle j \rangle M$	\rightarrow	$\langle i + j \rangle M$

Figure 1: Reduction rules of λ_l with de Bruijn indices

$$\begin{array}{c}
 \frac{}{_, A, \Delta \vdash \underline{i} : A} \text{ Axiom} \quad \frac{\Delta \vdash M : B}{_, \Delta \vdash \langle i \rangle M : B} \text{ Weak} \\
 \frac{_ \vdash M : B \rightarrow A \quad _ \vdash N : B}{_ \vdash (MN) : A} \text{ App} \quad \frac{B, _ \vdash M : C}{_ \vdash \lambda M : B \rightarrow C} \text{ Lambda} \\
 \\
 \frac{\Delta, \Pi \vdash N : A \quad _ , A, \Pi \vdash M : B}{_ , \Delta, \Pi \vdash [i/N, j]M : B} \text{ Subst}
 \end{array}$$

Figure 2: Typing rules for λ_l with de Bruijn indices

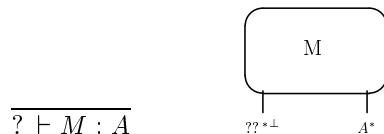
Theorem 4.1 (Subject Reduction) If $\Psi \vdash M : C$ and $M \rightarrow M'$, then $\Psi \vdash M' : C$.

4.2 Translation of types and terms of λ_l

We use the translation of types introduced in [6] given by :

$$\begin{aligned}
 A^* &= A && \text{if } A \text{ is an atomic type} \\
 (A \rightarrow B)^* &= ?((A^*)^\perp) \wp !B^* && \text{(that is, } !A^* \multimap !B^* \text{) otherwise}
 \end{aligned}$$

Since wires are commutative in proof nets, we feel free to exchange them when we define the translation of a term. The translation associates to every typed term M of λ_l , whose type judgment ends with the conclusion written below on the left, a proof net having the shape sketched below on the right:



Here is the formal definition of the translation T from λ_l -terms into proof nets.

- If the term is a variable and its type judgment ends with the rule written below on the left, then its translation is the proof net on the right

$$\frac{?, A, \Delta \vdash i : A}{Axiome} \quad \begin{array}{c} \text{Proof Net} \\ \text{Translation} \end{array}$$

$Axiome$

where i is the position of A in the typing environment,

- If the term is a λ -abstraction and its type judgment ends with the rule written below on the left, then its translation is the proof net on the right

$$\frac{B, ? \vdash M : C}{? \vdash \lambda M : B \rightarrow C} \quad \begin{array}{c} \text{Lambda} \\ \text{Proof Net} \end{array}$$

- If the term is an application and its type judgment ends with the rule written below on the left, then its translation is the proof net on the right

$$\frac{? \vdash M : B \rightarrow A \quad ? \vdash N : B}{? \vdash (MN) : A} \quad \begin{array}{c} \text{App} \\ \text{Proof Net} \end{array}$$

- If the term is a substitution and its type judgment ends with the rule written below on the left, then its translation is the proof net on the right

$$\frac{\Delta, \Pi \vdash N : A \quad ?, A, \Pi \vdash M : B}{?, \Delta, \Pi \vdash [i/N, j]M : B} \quad \begin{array}{c} \text{Subst} \\ \text{Proof Net} \end{array}$$

where i is the length of the list $?$ and j is the length of the list Δ , then its translation is the proof net

- Finally, if the term is a label and its type judgment ends with the rule written below on the left, then its translation is the proof net on the right

$$\frac{\Delta \vdash M : B}{?, \Delta \vdash \langle i \rangle M : B} \quad \begin{array}{c} \text{Weak} \\ \text{Proof Net} \end{array}$$

where i is the length of the list $?$, then its translation is the proof net

4.3 Simulating λ_l -reduction

We now verify that our notion of reduction R_E on PN simulates the λ_l -reduction on typed λ_l -terms. It is in this proof that we find the motivation for our choice of translation from λ -terms into proof nets: with the more traditional translation sending the intuitionistic type $A \rightarrow B$ into the linear $!A \multimap B$, the simulation of the rewrite rule f would give rise to an equality, not to a reduction step like in this paper.

Lemma 4.2 (Simulation of λ_l) The relation R_E simulates the λ_l -reduction on typed terms: if $t \rightarrow_{\lambda_l} t'$, then $T(t) \rightarrow^{+}_{R_E} T(t')$, excepted for the rules e_2 and d for which we have $T(t) = T(t')$.

Proof. The proof proceeds by cases on the reduction rule applied in the step $t \rightarrow_{\lambda_l} t'$. Since reductions λ_l and R_E are closed under all contexts, we only need to study the cases where reduction takes place at the head position of t . In the proof, rule wc is used to simulate b_2, e_1, n_1, n_2, n_3 , equivalence A is used to simulate a, c_1, c_2 , and equivalence B is used to simulate f, a, c_1, c_2 .

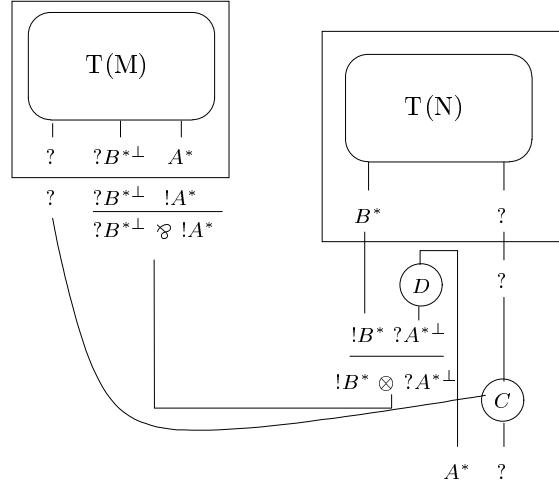
- **rule b_1** : $(\lambda MN) \rightarrow [0/N, 0]M$

The typing judgment of (λMN) ends with

$$\frac{B, ? \vdash M : A}{? \vdash \lambda M : B \rightarrow A} \text{ Lambda} \quad ? \vdash N : B \text{ App}$$

$$? \vdash (\lambda MN) : A$$

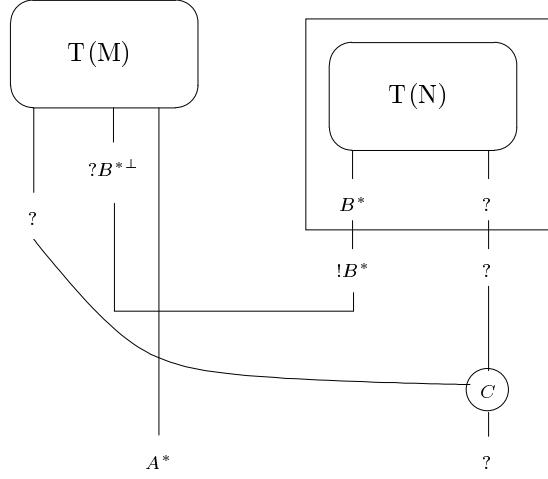
and its translation is the proof net



The typing judgment of $[0/N, 0]M$ must end with:

$$\frac{B, ? \vdash M : A \quad ? \vdash N : B}{? \vdash [0/N, 0]M : A} \text{ Subst}$$

and its translation is the proof net



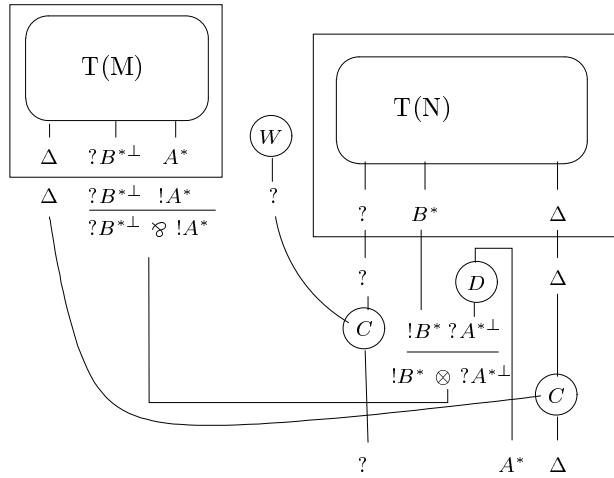
Starting from the first proof net, we eliminate the $\wp\text{-}\otimes$ cut, then the $d - b$ cut and finally the $Ax - cut$ to obtain the final proof net.

- rule $b_2 : ((\langle k \rangle \lambda M)N) \rightarrow [0/N, k]M$

The typing environment can be split in two parts Δ and Δ , where k is the length of Δ . The typing judgment of $((\langle k \rangle \lambda M)N)$ ends with

$$\frac{\frac{B, \Delta \vdash M : A}{\Delta \vdash \lambda M : B \rightarrow A} \quad ? , \Delta \vdash N : B}{? , \Delta \vdash ((\langle k \rangle \lambda M)N) : A}$$

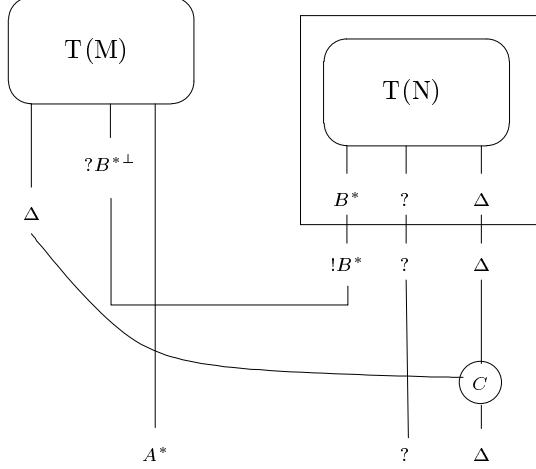
and its translation is the proof net



The typing judgment of $[0/N, k]M$ must end with:

$$\frac{? , \Delta \vdash N : B \quad B, \Delta \vdash M : A}{? , \Delta \vdash [0/N, k]M : A} Subst$$

and its translation is the proof net



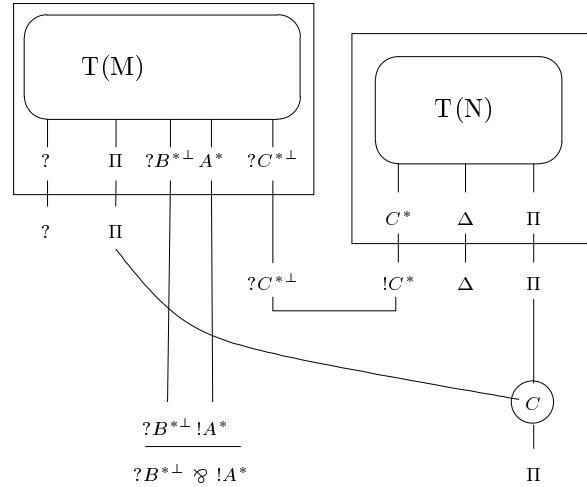
As for the $b - 1$ rule, we eliminate the $\wp\text{-}\otimes$ cut, then the $d - b$ cut, and the $Ax - cut$. Finally, we apply the wc rule to achieve the desired result.

- rule $f : [i/N, j]\lambda M \rightarrow \lambda[i + 1/N, j]M$

The typing environment can be split in three parts $?, \Delta, \Pi$, where i is the length of $?$ and j is the length of Δ . The typing judgment of $[i/N, j]\lambda M$ ends with

$$\frac{\Delta, ?, C, \Pi \vdash M : A \quad \Delta, \Pi \vdash N : C \quad \frac{?, C, \Pi \vdash \lambda M : B \rightarrow A}{?, \Delta, \Pi \vdash [i/N, j]\lambda M : B \rightarrow A} \text{ Lambda}}{?, \Delta, \Pi \vdash [i/N, j]\lambda M : B \rightarrow A} \text{ Subst}$$

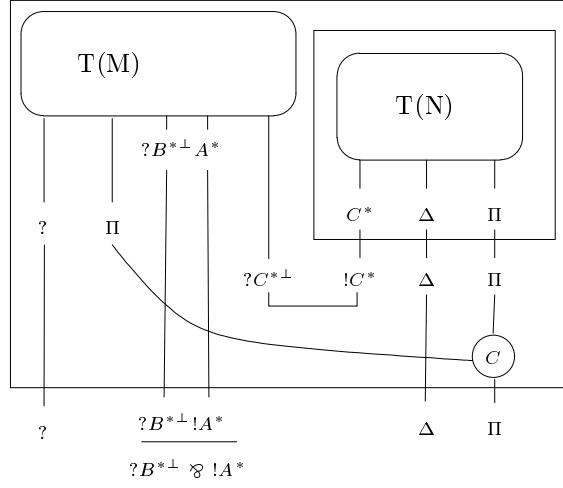
and its translation is the proof net



The typing judgment of $\lambda[i + 1/N, j]M$ must end with:

$$\frac{\Delta, \Pi \vdash N : C \quad B, ?, C, \Pi \vdash M : A}{B, ?, \Delta, \Pi \vdash [i + 1/N, j]M : A} \text{ Subst} \quad \frac{B, ?, \Delta, \Pi \vdash [i + 1/N, j]M : A}{?, \Delta, \Pi \vdash \lambda[i + 1/N, j]M : B \rightarrow A} \text{ Lambda}$$

and its translation is the proof net



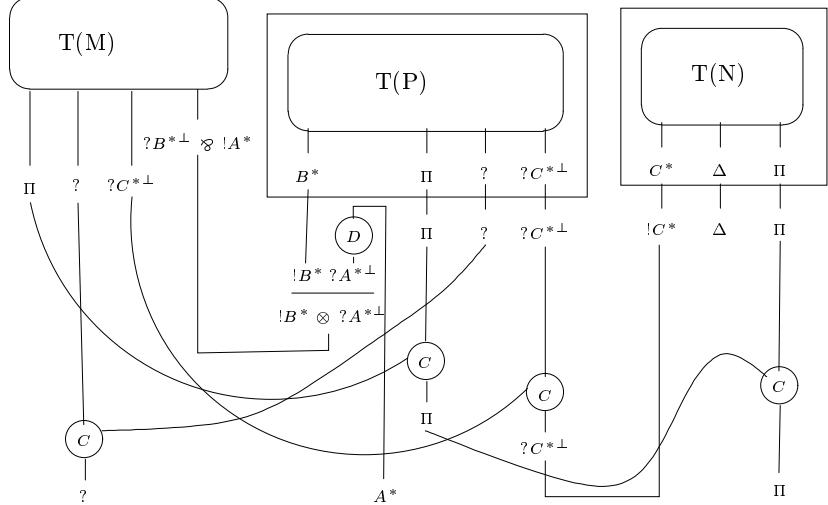
To reduce the first proof net into the second one, we must eliminate the $b - b$ cut, then use the equivalence relation B (we will exactly show how to use the equivalence relations in the case of the rule a).

- **rule a** : $[i/N, j](MP) \longrightarrow (([i/N, j]M)([i/N, j]P))$

The typing environment can be split in three parts $?, \Delta, \Pi$, where i is the length of $?$ and j is the length of Δ . The typing judgment of $[i/N, j](MP)$ ends with

$$\frac{\Delta, \Pi \vdash N : C \quad \frac{?, C, \Pi \vdash M : B \rightarrow A \quad ?, C, \Pi \vdash P : B}{?, C, \Pi \vdash MP : A} App}{?, \Delta, \Pi \vdash [i/N, j](MP) : A} Subst$$

and its translation is the proof net

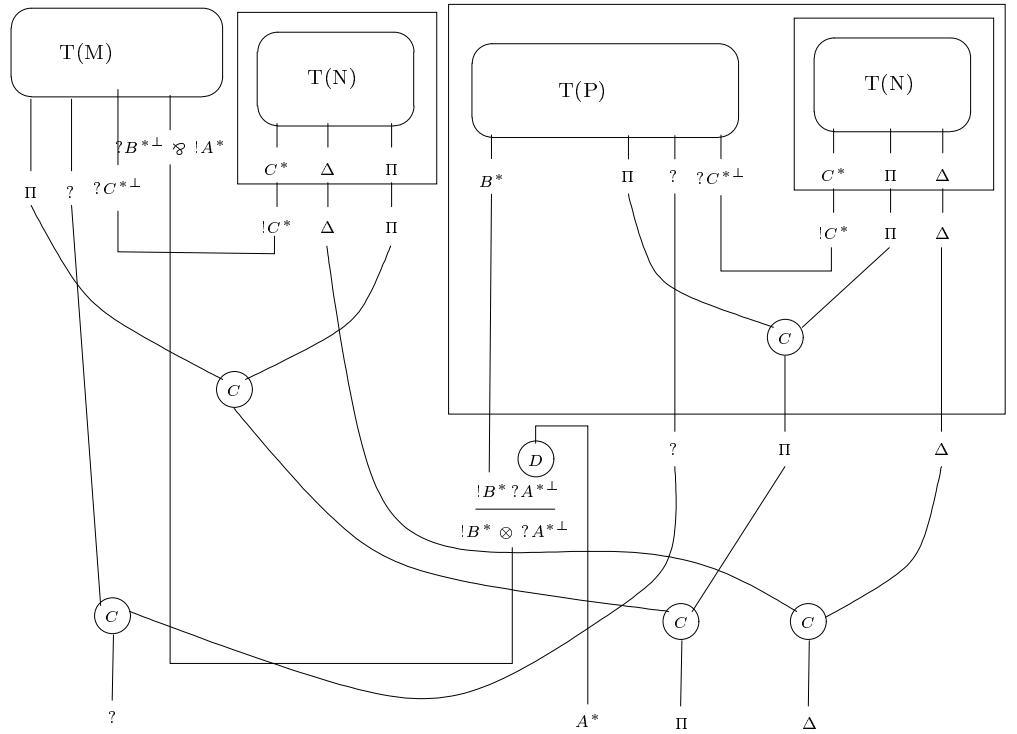


The typing judgment of $(([i/N, j]M)([i/N, j]P))$ must end with:

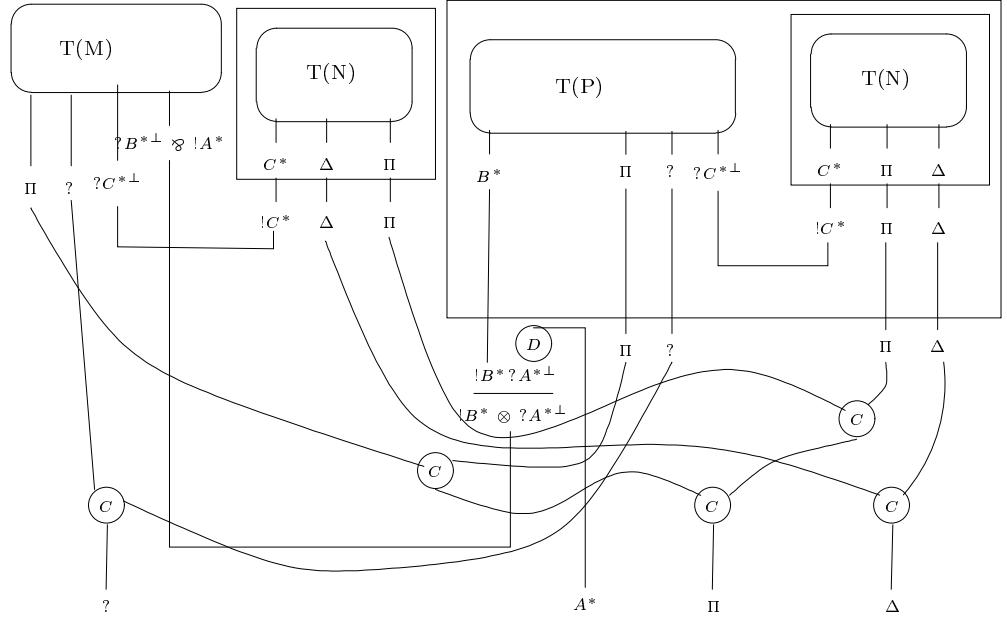
$$\frac{\Delta, \Pi \vdash N : C \quad ?, C, \Pi \vdash M : B \rightarrow A}{?, \Delta, \Pi \vdash ([i/N, j]M) : B \rightarrow A} Subst \quad \frac{\Delta, \Pi \vdash N : C \quad ?, C, \Pi \vdash P : B}{?, \Delta, \Pi \vdash ([i/N, j]P) : B} Subst$$

$$?, \Delta, \Pi \vdash (([i/N, j]M)([i/N, j]P)) : A \quad App$$

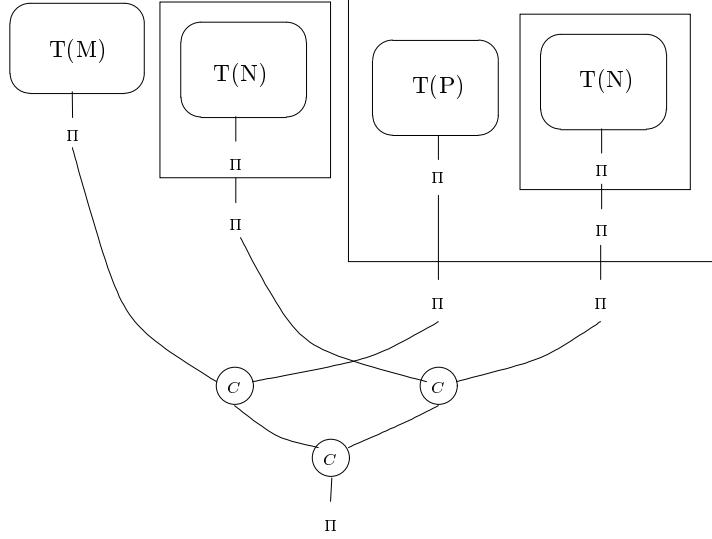
and its translation is the proof net



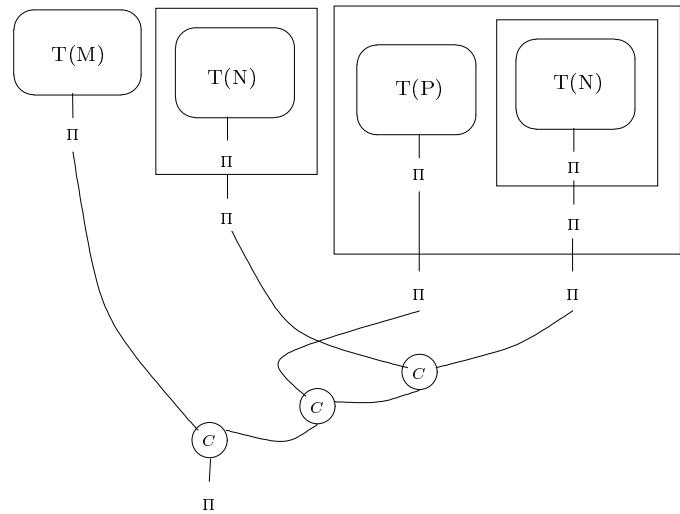
We eliminate the $c - b$ cut, then the $b - b$ cut, and thus we get the following proof net:



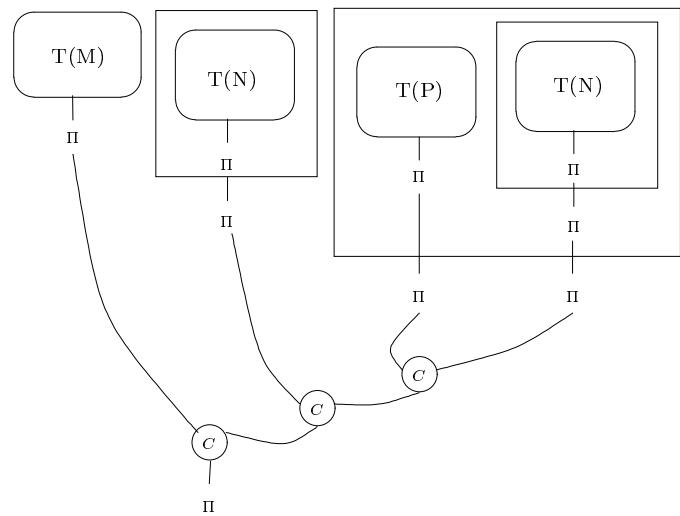
To get to the desired proof net we need to use the equivalence relations A and B which were introduced in Section 2. To better understand how to use them, we focus on the crucial informations, i.e. the contraction nodes and their connections with the nets $T(M)$, $T(N)$ and $T(P)$. Here is the net corresponding to the above net :



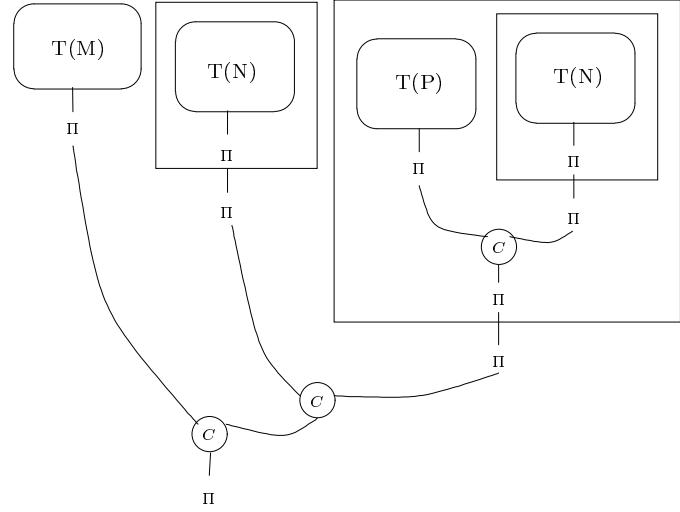
We use the associativity axiom A to obtain :



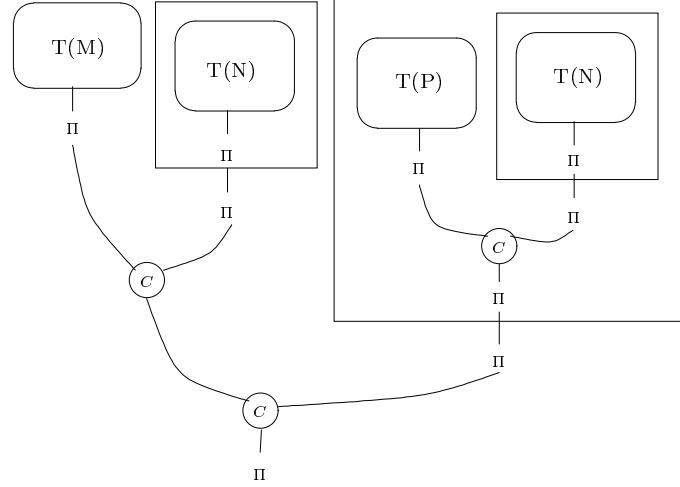
Again by associativity we get



Using the B axiom we can put the contraction inside the box :



And finally we use the A axiom again to obtain the desired proof net :

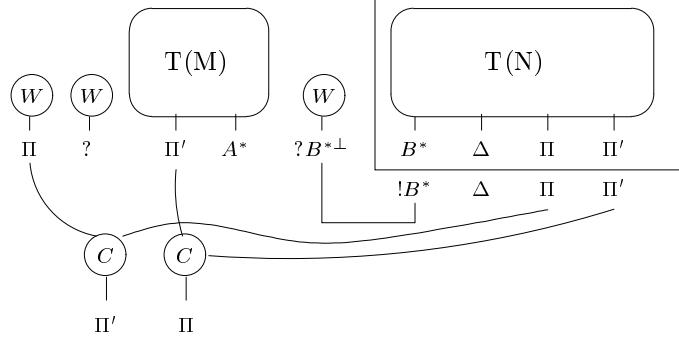


- rule $e_1 : [i/N, j]\langle k \rangle M \rightarrow \langle j + k - 1 \rangle M$ if $i < k$

The typing environment can be split in four parts $?$, Δ , Π , and Π' , where i is the length of $?$, j is the length of Δ , and k ($k > i$) is the length of $?$ plus the length of Π plus 1. The typing judgment of $[i/N, j]\langle k \rangle M$ ends with

$$\frac{\Delta, \Pi, \Pi' \vdash N : B \quad \frac{\Pi' \vdash M : A}{?, B, \Pi, \Pi' \vdash \langle k \rangle M : A} \text{ Weak}}{?, \Delta, \Pi, \Pi' \vdash [i/N, j]\langle k \rangle M : A} \text{ Subst}$$

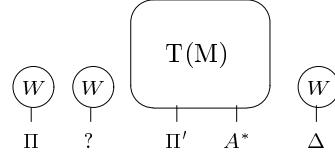
and its translation is the proof net



The typing judgment of $\langle j+k-1 \rangle M$ must end with:

$$\frac{\Pi' \vdash M : A}{?, \Delta, \Pi, \Pi' \vdash \langle j+k-1 \rangle M : A} \text{ Weak}$$

and its translation is the net



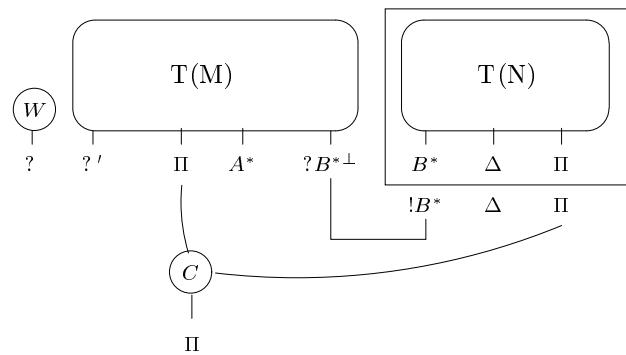
Starting from the first proof net, we eliminate the $w - b$ cut, then we apply the wc rule and we finally obtain the desired proof net.

- **rule e_2** : $[i/N, j]\langle k \rangle M \rightarrow \langle k \rangle [i-k/N, j]M$ if $i \geq k$

The typing environment can be split in four parts $?, ?, \Delta, \Pi$, where i is the length of $?$ plus the length of $?'$, j is the length of Δ and k ($k \leq i$) is the length of $?$. The typing judgment of $[i/N, j]\langle k \rangle M$ ends with

$$\frac{\Delta, \Pi \vdash N : B \quad \frac{?, ?, B, \Pi \vdash M : A}{?, ?, \Delta, \Pi \vdash [i/N, j]\langle k \rangle M : A} \text{ Subst}}{?, ?, \Delta, \Pi \vdash [i/N, j]\langle k \rangle M : A} \text{ Weak}$$

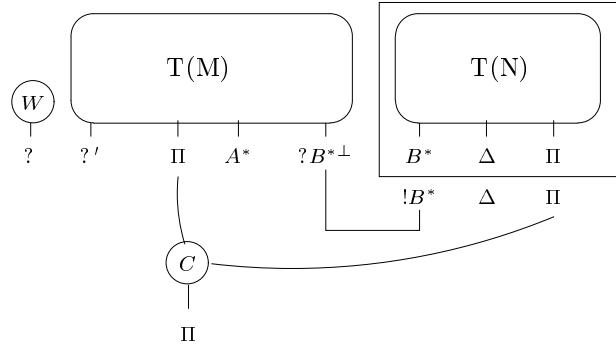
and its translation is the proof net



The typing judgment of $\langle k \rangle [i - k/N, j]M$ must end with:

$$\frac{\frac{\Delta, \Pi \vdash N : B \quad ?', B, \Pi \vdash M : A}{? ', \Delta, \Pi \vdash [i - k/N, j]M : A} Subst}{?, ?, \Delta, \Pi \vdash \langle k \rangle [i - k/N, j]M : A} Weak$$

and its translation is the proof net



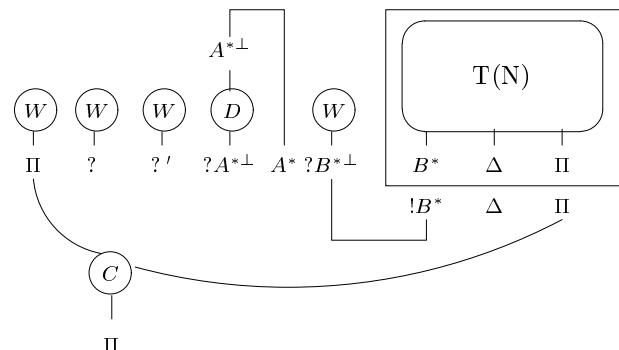
We notice that the two nets are already the same. This is the first of the exception cases of the lemma.

- rule $n_1 : [i/N, j]\underline{k} \longrightarrow \underline{k}$ if $i > k$

The typing environment can be split in five parts $?, A, ?, ', \Delta, \Pi$, where i is the length of $?$ plus the length of $?'$ plus 1, j is the length of Δ and k ($k < i$) is the length of $?$. The typing judgment of $[i/N, j]\underline{k}$ ends with

$$\frac{\Delta, \Pi \vdash N : B \quad \overline{?, A, ?, ', B, \Pi \vdash \underline{k} : A} \quad Axiome}{?, A, ?, ', \Delta, \Pi \vdash [i/N, j]\underline{k} : A} Subst$$

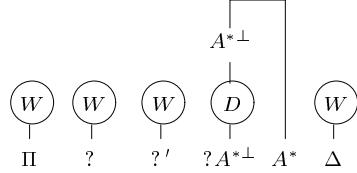
and its translation is the proof net



The typing judgment of \underline{k} must end with:

$$\overline{?, A, ?, ', \Delta, \Pi \vdash \underline{k} : A}$$

and its translation is the proof net



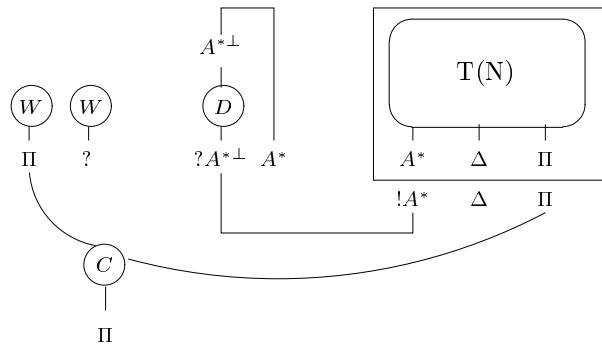
To reduce the first proof net into the second one it is enough to eliminate the $w - b$ cut and to apply the wc rule.

- **rule n_2** : $[i/N, j]\underline{i} \rightarrow \langle i\rangle N$

The typing environment can be split in three parts $\underline{?}$, Δ , Π , where i is the length of $\underline{?}$ and j is the length of Δ . The typing judgment of $[i/N, j]\underline{i}$ ends with

$$\frac{\Delta, \Pi \vdash N : A \quad \overline{?, A, \Pi \vdash \underline{i} : A} \quad \text{Axiome}}{?, \Delta, \Pi \vdash [i/N, j]\underline{i} : A} \text{ Subst}$$

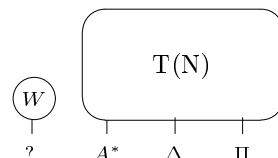
and its translation is the proof net



The typing judgment of $\langle i\rangle N$ must end with:

$$\frac{\Delta, \Pi \vdash N : A}{?, \Delta, \Pi \vdash \langle i\rangle N : A} \text{ Weak}$$

and its translation is the proof net



Starting from the first proof net, we eliminate the $d - b$ cut, then the $Ax - cut$ cut, and we apply the wc rule to obtain the desired proof net.

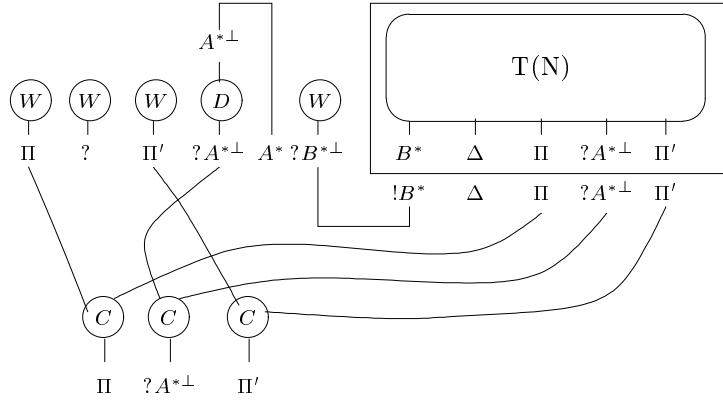
- **rule n_3** : $[i/N, j]\underline{k} \rightarrow \underline{j+k-1}$ if $i < k$

The typing environment can be split in five parts $\underline{?}$, Δ , Π , A , Π' , where i is the length of $\underline{?}$, j is the length of Δ and k ($k > i$) is the length of $\underline{?}$ plus the length of Π plus 1.

The typing judgment of $[i/N, j]\underline{k}$ ends with

$$\frac{\Delta, \Pi, A, \Pi' \vdash N : B \quad ?, B, \Pi, A, \Pi' \vdash \underline{k} : A}{?, \Delta, \Pi, A, \Pi' \vdash [i/N, j]\underline{k} : A} \text{ Axiome } Subst$$

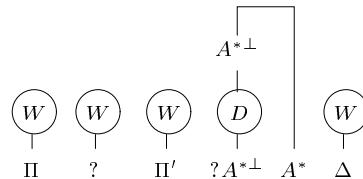
and its translation is the proof net



The typing judgment of j+k-1 must end with:

$$\frac{?, \Delta, \Pi, A, \Pi' \vdash \underline{j+k-1} : A}{?, \Delta, \Pi, A, \Pi' \vdash \underline{j+k-1} : A} \text{ Axiome }$$

and its translation is the proof net



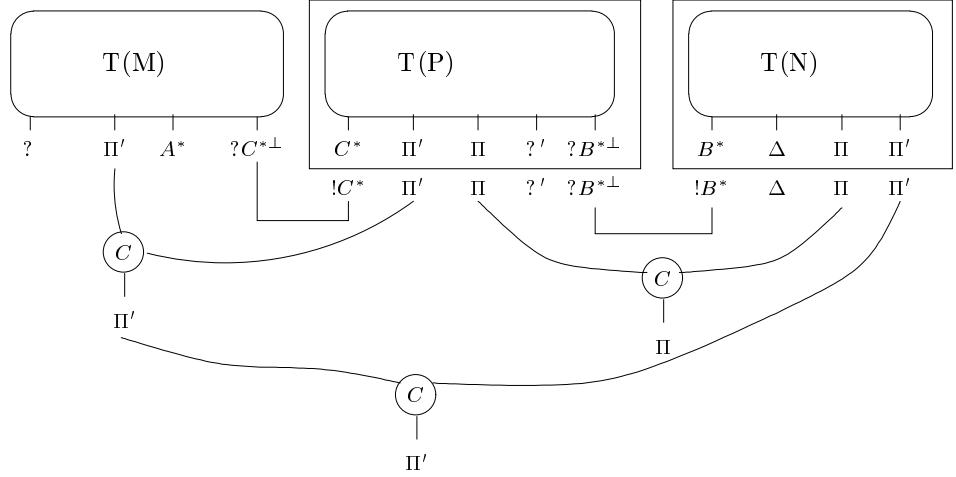
As for the n_1 rule, we eliminate the $w - b$ cut, then we apply three times the wc rule to achieve the desired result.

- **rule c_1** : $[i/N, j][k/P, l]M \rightarrow [k/[i-k/N, j]P, j+l-1]M$ if $k \leq i < k+l$

The typing environment can be split into five parts $?, ?, \Delta, \Pi, \Pi'$, where i is the length of $?$ plus the length of $?'$, j is the length of Δ , k ($k \leq i$) is the length of $?$ and l ($k+l > i$) is the length of $?'$ plus the length of Π plus 1. The typing judgment of $[i/N, j][k/P, l]M$ ends with

$$\frac{\Delta, \Pi, \Pi' \vdash N : B \quad ?, ?, B, \Pi, \Pi' \vdash [k/P, l]M : A}{?, ?, \Delta, \Pi, \Pi' \vdash [i/N, j][k/P, l]M : A} \text{ Subst }$$

and its translation is the proof net

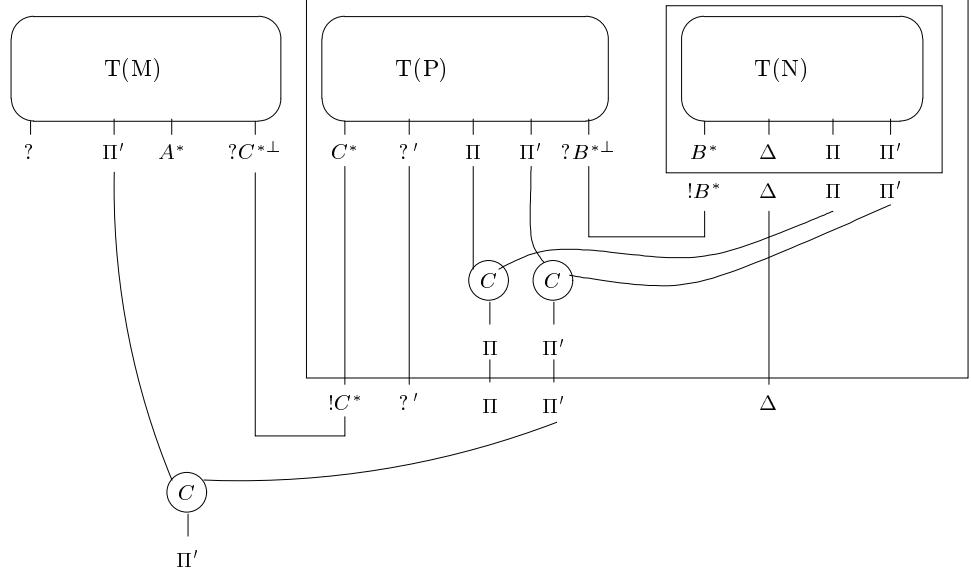


The typing judgment of $[k/[i - k/N, j]P, j + l - 1]M$ must end with:

$$\frac{\Delta, \Pi, \Pi' \vdash N : B \quad ?', B, \Pi, \Pi' \vdash P : C}{? ', \Delta, \Pi, \Pi' \vdash [i - k/N, j]P : C} \text{ Subst} \quad ?, C, \Pi' \vdash M : A \quad \text{Subst}$$

$$?, ?, \Delta, \Pi, \Pi' \vdash [k/[i - k/N, j]P, j + l - 1]M : A$$

and its translation is the proof net



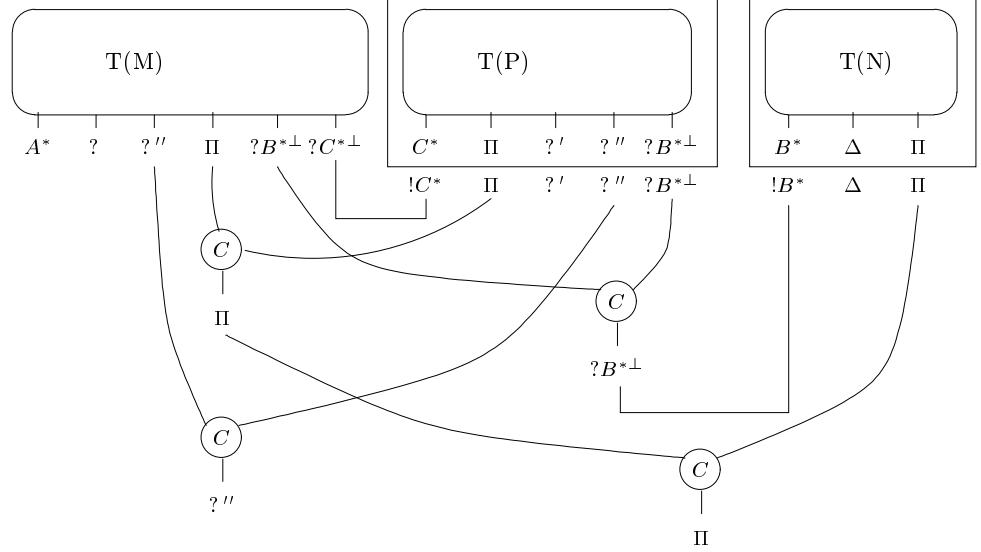
To reduce the first proof net into the second one, we must eliminate the $b - b$ cut, then apply the equivalence relations A and B .

- rule $c_2 : [i/N, j][k/P, l]M \rightarrow [k/[i - k/N, j]P, l][i - l + 1/N, j]M$ if $k + l \leq i$

The typing environment can be split in five parts $?, ?, ?, \Delta, \Pi$, where i is the length of $?$ plus the length of $?'$ plus the length of $?''$, j is the length of Δ , k ($k + l \leq i$) is the length of $?$ and l is the length of $?'$. The typing judgment of $[i/N, j][k/P, l]M$ ends with

$$\frac{\Delta, \Pi \vdash N : B \quad ?', ?, B, \Pi \vdash P : C \quad ?, C, ?, B, \Pi \vdash M : A}{\begin{array}{c} ?, ?, ?, B, \Pi \vdash [k/P, l]M : A \\ ?, ?, ?, \Delta, \Pi \vdash [i/N, j][k/P, l]M : A \end{array}} \text{Subst}$$

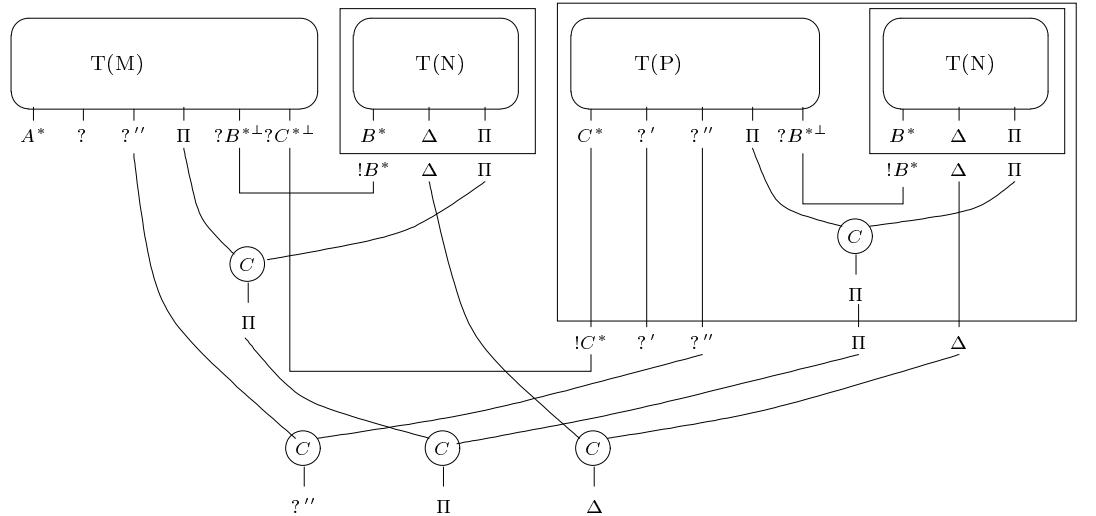
and its translation is the proof net



The typing judgment of $[k/[i - k/N, j]P, l][i - l + 1/N, j]M$ must end with:

$$\frac{\Delta, \Pi \vdash N : B \quad ?', ?, B, \Pi \vdash P : C \quad \Delta, \Pi \vdash N : B \quad ?, C, ?, B, \Pi \vdash M : A}{\begin{array}{c} ?, ?, ?, \Delta, \Pi \vdash [i - k/N, j]P : C \\ ?, C, ?, \Delta, \Pi \vdash [i - l + 1/N, j]M : A \end{array}} \text{Subst}$$

and its translation is the proof net



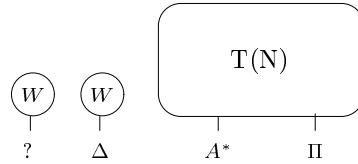
Starting from the first proof net, we eliminate the $c - b$ cut, then the $b - b$ cut, and we apply the equivalence rules A and B to obtain the desired proof net.

- **rule d** : $\langle i \rangle \langle j \rangle M \longrightarrow \langle i + j \rangle M$

The typing environment can be split in three parts $?, \Delta, \Pi$, where i is the length of $?$ and j is the length of Δ . The typing judgment of $\langle i \rangle \langle j \rangle M$ ends with

$$\frac{\frac{\Pi \vdash M : A}{\Delta, \Pi \vdash \langle j \rangle M : A} \text{ Weak}}{?, \Delta, \Pi \vdash \langle i \rangle \langle j \rangle M : A} \text{ Weak}$$

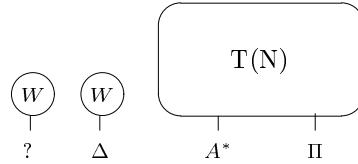
and its translation is the proof net



The typing judgment of $\langle i + j \rangle M$ must end with:

$$\frac{\Pi \vdash M : A}{?, \Delta, \Pi \vdash \langle i + j \rangle M : A} \text{ Weak}$$

and its translation is the proof net



We notice that the two proof nets are already the same. This is the second of the exception cases of the lemma.

4.4 The proof of strong normalization of λ_l

We are now able to show strong normalization of λ_l . To achieve this result, we use the following abstract theorem (see for example [11]) :

Theorem 4.3 Let $R = \langle \mathcal{O}, R_1 \cup R_2 \rangle$ be an abstract reduction system such that R_2 is strongly normalizing and there exist a reduction system $S = \langle \mathcal{O}', R' \rangle$, with a translation T of \mathcal{O} into \mathcal{O}' such that $a \rightarrow_{R_1} b$ implies $T(a) \rightarrow^{+}_{R'} T(b)$; $a \rightarrow_{R_2} b$ implies $T(a) = T(b)$. Then if R' is strongly normalizing, $R_1 \cup R_2$ is also strongly normalizing.

If we take \mathcal{O} as the set of typed λ_l -terms, R_1 as $\lambda_l - \{e_2, d\}$, R_2 as $\{e_2, d\}$, \mathcal{O}' as the set of proof nets and R' as the reduction R_E , then, by the Theorem 4.3 and the fact that the system including the rules $\{e_2, d\}$ is strongly normalizing [8], we can conclude :

Theorem 4.4 (Strong normalization of λ_l) The typed λ_l -calculus is strongly normalizing.

5 The λ_l -calculus with names

In this section we present a version of typed λ_l with named variables. We first introduce the grammar of terms, then the typing and reduction rules, and finally, we will briefly discuss the translation of this syntax to PN .

The terms of this calculus are given by the following grammar:

$M ::=$	x	<i>variable</i>
	$\lambda x.M$	<i>abstraction</i>
	(MM)	<i>application</i>
	ΔM	<i>label</i>
	$M[x, M, ?, \Delta]$	<i>substitution</i>

Intuitively, the term ΔM means that the variables in Δ are not in M , and the term $M[x, N, ?, \Delta]$ means that the variables in $?$ do not appear in N (they only belong to the type environment of M) and the variables Δ do not appear in M (they only belong to the type environment of N).

Variables are bound by the abstraction and substitution operators, so that for example x is bound in $\lambda x.x$ and in $x[x, N, ?, \Delta]$.

Terms are identified modulo α -conversion so that bound variables can be systematically renamed. Indeed, we have $\lambda y.y[x, z, \emptyset, \emptyset] =_{\alpha} \lambda y'.y'[x, z, \emptyset, \emptyset]$ and $\lambda y.y[x, z, \emptyset, \emptyset] =_{\alpha} \lambda y.y[x', z, \emptyset, \emptyset]$ and $\lambda l.y[x, z, \{l\}, \emptyset] =_{\alpha} \lambda l'.y[x, z, \{l'\}, \emptyset]$. We remark that the conditions on indices used in the typing rules given in Section 4.1 are now conditions on sets of variables. The typing rules are given in Figure 3.

$$\begin{array}{c}
 \frac{}{?, x : A \vdash x : A} \text{Axiom} \quad \frac{? \vdash M : A \quad ? \cap \Delta = \emptyset}{?, \Delta \vdash \Delta M : A} \text{Weak} \\
 \frac{? \vdash M : B \rightarrow A \quad ? \vdash N : B}{? \vdash (MN) : A} \text{App} \quad \frac{?, x : A \vdash M : B}{? \vdash \lambda x : A. M : B \rightarrow A} \text{Lambda} \\
 \frac{\Delta, \Pi \vdash N : A \quad ?, x : A, \Pi \vdash M : B \quad (? , x : A) \cap \Delta = \emptyset}{\Delta, ?, \Pi \vdash M[x, N, ?, \Delta] : B} \text{Subst}
 \end{array}$$

Figure 3: Typing rules for the λ_l -calculus with named variables

We remark that whenever $? \vdash M[x, N, \Delta, \Pi]$ is derivable, then $?$ necessarily contains Δ and Π .

As expected the λ_l -calculus with names enjoys the subject reduction property.

Theorem 5.1 (Subject Reduction) If $\Psi \vdash M : C$ and $M \rightarrow M'$, then $\Psi \vdash M' : C$.

Proof. By induction on the structure of terms. If the reduction takes place in an internal position of M , then we can apply the induction hypothesis. Otherwise, we verify all the possible cases.

- Rule b_1 .

$$\frac{\Psi, x : A \vdash M : C \quad \Psi \vdash N : A}{\Psi \vdash (\lambda x : A. M)N : C} \text{(app)} \quad \frac{\Psi, x : A \vdash M : C}{\Psi \vdash \lambda x : A. M : A \rightarrow C} \text{(lamb)}$$

$$\frac{\Psi, x : A \vdash M : C \quad \Psi \vdash N : A}{\Psi \vdash M[x, N, \emptyset, \emptyset] : C} (sub)$$

- Rule b_2 . Without loss of generality we can suppose that $\Psi = ?, \Lambda$.

$$\frac{\frac{\frac{?, x : A \vdash M : C}{?, \vdash \lambda x : A.M : A \rightarrow C} (lamb) \quad \Lambda = \Delta \quad ? \cap \Lambda = \emptyset}{?, \Lambda \vdash \Delta(\lambda x : A.M) : A \rightarrow C} (weak) \quad ?, \Lambda \vdash N : A}{?, \Lambda \vdash (\Delta(\lambda x : A.M))N : C} (app)$$

Since x is bound in $\lambda x : A.M$ we can suppose that $\Lambda = \Delta$ does not contain x , so that we obtain

$$\frac{?, x : A \vdash M : C \quad ?, \Lambda \vdash N : A \quad x : A \cap \Lambda = \emptyset \quad \Lambda = \Delta}{?, \Lambda \vdash M[x, N, \emptyset, \Delta] : C} (sub)$$

- Rule f . Without loss of generality we can suppose that $\Psi = ?, \Delta, \Pi$ and $C = A \rightarrow C'$.

$$\frac{\Delta, \Pi \vdash N : B \quad \frac{?, \Pi, x : B, y : A \vdash M : C'}{?, \Pi, x : B \vdash \lambda y : A.M : C} (lamb) \quad (? , x : B) \cap \Delta = \emptyset}{?, \Delta, \Pi \vdash (\lambda y : A.M)[x, N, ?, \Delta] : C} (sub)$$

Since y is bound in $\lambda y : A.M$ we can suppose that Δ does not contain y , so that we obtain

$$\frac{\Delta, \Pi \vdash N : B \quad ?, \Pi, x : B, y : A \vdash M : C' \quad (? , x : B, y : A) \cap \Delta = \emptyset}{\frac{?, \Delta, \Pi, y : A \vdash M[x, N, (? , x : B), \Delta] : C'}{?, \Delta, \Pi \vdash \lambda y : A.M[x, N, (? , x : B), \Delta] : C}} (lamb)$$

- Rule a . Without loss of generality we can suppose that $\Psi = ?, \Delta, \Pi$.

$$\frac{\Delta, \Pi \vdash N : B \quad \frac{?, \Pi, x : B \vdash M : A \rightarrow C \quad ?, \Pi, x : B \vdash P : A}{?, \Pi, x : B \vdash (MP) : C} (app) \quad (? , x : B) \cap \Delta = \emptyset}{?, \Delta, \Pi \vdash (MP)[x, N, ?, \Delta] : C} (sub)$$

$$\frac{\Delta, \Pi \vdash N : B \quad ?, \Pi, x : B \vdash M : A \rightarrow C \quad (? , x : B) \cap \Delta = \emptyset}{?, \Delta, \Pi \vdash M[x, N, ?, \Delta] : A \rightarrow C} (sub)$$

$$\frac{\Delta, \Pi \vdash N : B \quad ?, \Pi, x : B \vdash P : A \quad (? , x : B) \cap \Delta = \emptyset}{?, \Delta, \Pi \vdash P[x, N, ?, \Delta] : A} (sub)$$

$$\frac{?, \Delta, \Pi \vdash M[x, N, ?, \Delta] : A \rightarrow C \quad ?, \Delta, \Pi \vdash P[x, N, ?, \Delta] : A}{?, \Delta, \Pi \vdash (M[x, N, ?, \Delta]P[x, N, ?, \Delta]) : C} (app)$$

- Rule e_1 . Without loss of generality we can suppose that $\Psi = ?, \Delta, \Pi$. We know also, by hypothesis, that $x \in \Lambda$, so that Λ can be written as $x : B, ?, \Pi'', \Pi'$, where $? = ?, ?, \Pi = \Pi', \Pi''$ and such that the set difference $? \setminus \Lambda$ is $?'$ and $\Pi \setminus \Lambda$ is Π' .

$$\frac{\Delta, \Pi \vdash N : B \quad ?', \Pi' \vdash M : C \quad \Lambda \cap (?', \Pi') = \emptyset \quad (weak) \quad (? , x : B) \cap \Delta = \emptyset \quad (sub)}{?, \Delta, \Pi \vdash \Lambda M[x, N, ?, \Delta] : C}$$

Since $\Pi' \subseteq \Pi$, then it is evident that $\Delta \cap \Pi' = \emptyset$, and since $?' \subseteq ?$, then $\Delta \cap ?' = \emptyset$ comes from the fact that $(?, x : B) \cap \Delta = \emptyset$. Indeed, $(\Lambda \setminus x) \cap (?', \Pi') = \emptyset$ is a consequence of $\Lambda \cap (?', \Pi') = \emptyset$. We thus obtain

$$\frac{?', \Pi' \vdash M : C \quad (\Delta \cup (\Lambda \setminus x)) \cap (?', \Pi') = \emptyset \quad (weak)}{?, \Delta, \Pi \vdash (\Delta \cup (\Lambda \setminus x))M : C}$$

- Rule e_2 . Without loss of generality we can suppose that $\Psi = ?, \Delta, \Pi$. We also know, by hypothesis, that $x \notin \Lambda$, so that Λ can be written as $?, \Pi'', \Pi''$, where $? = ?, ?, \Pi = \Pi', \Pi''$ and so $?', \Pi', x : B, \Lambda = ?, \Pi, x : B$.

$$\frac{\Delta, \Pi \vdash N : B \quad ?', \Pi', x : B \vdash M : C \quad (?', \Pi', x : B) \cap \Lambda = \emptyset \quad (weak) \quad (? , x : B) \cap \Delta = \emptyset \quad (sub)}{?, \Delta, \Pi \vdash \Lambda M[x, N, ?, \Delta] : C}$$

$$\frac{\Delta, \Pi \vdash N : B \quad ?', \Pi', x : B \vdash M : C \quad ?' = ? \setminus \Lambda \quad \Pi'' = \Lambda \setminus ? \quad (sub) \quad (weak)}{?, \Delta, \Pi, ?' \vdash M[x, N, ? \setminus \Lambda, \Delta \cup (\Lambda \setminus ?)] : C} \quad (weak)$$

- Rule n_1 . Without loss of generality we can suppose that $\Psi = ?, \Delta, \Pi$.

$$\frac{\Delta, \Pi \vdash N : B \quad ? , \Pi, x : B \vdash y : C \quad (ax) \quad (? , x : B) \cap \Delta = \emptyset \quad (sub)}{?, \Delta, \Pi \vdash y[x, N, ?, \Delta] : C}$$

$$\frac{}{?, \Delta, \Pi, x : B \vdash y : C} (ax)$$

- Rule n_2 . Without loss of generality we can suppose that $\Psi = ?, \Delta, \Pi$.

$$\frac{\Delta, \Pi \vdash N : C \quad ? , \Pi, x : C \vdash x : C \quad (? , x : C) \cap \Delta = \emptyset \quad (sub)}{?, \Delta, \Pi \vdash y[x, N, ?, \Delta] : C}$$

$$\frac{\Delta, \Pi \vdash N : C \quad ? \cap \Delta = \emptyset \quad (weak)}{?, \Delta, \Pi \vdash ?N : C}$$

- Rule c_1 . Without loss of generality we can suppose that $\Psi = ?, \Delta, \Pi$, where $? = \Phi', \Lambda, \Omega$. Since $x \in \Phi \setminus \Lambda$, then Φ can be written as $\Phi', x : B$.

$$\frac{\Delta, \Pi \vdash N : B \quad \frac{\Phi', \Omega, x : B, \Pi \vdash P : C \quad \Lambda, y : A, \Omega, \Pi \vdash M : C \quad (\Lambda, y : A) \cap (\Phi', x : B) = \emptyset}{\Phi', \Lambda, \Omega = ?, x : B, \Pi \vdash M[y, P, \Lambda, \Phi] : C} (sub) \quad (? , x : B) \cap \Delta = \emptyset}{? , \Delta, \Pi \vdash M[y, P, \Lambda, \Phi][x, N, ?, \Delta] : C}$$

$$\frac{\Delta, \Pi \vdash N : B \quad x : B, \Phi', \Omega, \Pi \vdash P : C \quad (x : B, \Phi', \Omega) \cap \Delta = \emptyset}{\Phi', \Omega, \Delta, \Pi \vdash P[x, N, ? \setminus \Lambda, \Delta] : C} (sub)$$

Since y is bound in $M[y, P, \Lambda, \Phi]$, then we can suppose that Δ does not contain y , so that we obtain

$$\frac{\Phi', \Omega, \Delta, \Pi \vdash P[x, N, ? \setminus \Lambda, \Delta] : C \quad \Lambda, y : A, \Omega, \Pi \vdash M : C \quad (y : A, \Lambda) \cap (\Delta, \Phi') = \emptyset}{\Phi', \Lambda, \Omega, \Delta, \Pi \vdash M[y, P[x, N, ? \setminus \Lambda, \Delta], \Lambda, \Delta \cup (\Phi \setminus x)] : C} (sub)$$

- Rule c_2 . Without loss of generality we can suppose that $\Psi = ?, \Delta, \Pi$, where $? = \Phi, \Lambda, \Omega$. We also have $x \notin \Phi \cup \Lambda$.

$$\frac{\Delta, \Pi \vdash N : B \quad \frac{\Phi, \Omega, x : B, \Pi \vdash P : C \quad \Lambda, y : A, \Omega, x : B, \Pi \vdash M : C \quad (\Lambda, y : A) \cap \Phi = \emptyset}{\Phi, \Lambda, \Omega = ?, x : B, \Pi \vdash M[y, P, \Lambda, \Phi] : C} (sub) \quad (? , x : B) \cap \Delta = \emptyset}{? , \Delta, \Pi \vdash M[y, P, \Lambda, \Phi][x, N, ?, \Delta] : C} (sub)$$

$$\frac{\Delta, \Pi \vdash N : B \quad \Phi, \Omega, x : B, \Pi \vdash P : C \quad (\Phi, \Omega, x : B) \cap \Delta = \emptyset}{\Phi, \Omega, \Delta, \Pi \vdash P[x, N, ? \setminus \Lambda, \Delta] : C} (sub)$$

Since y is bound in $M[y, P, \Lambda, \Phi]$, then we can suppose that Δ does not contain y , so that we obtain

$$\frac{\Delta, \Pi \vdash N : B \quad \Lambda, y : A, \Omega, x : B, \Pi \vdash M : C \quad (\Lambda, y : A, \Omega, x : B) \cap \Delta = \emptyset}{\Lambda, y : A, \Omega, \Delta, \Pi \vdash M[x, N, (? \setminus \Phi) + y, \Delta] : C} (sub)$$

$$\frac{\Phi, \Omega, \Delta, \Pi \vdash P[x, N, ? \setminus \Lambda, \Delta] : C \quad \Lambda, y : A, \Omega, \Delta, \Pi \vdash M[x, N, (? \setminus \Phi) + y, \Delta] : C \quad (\Lambda, y : A) \cap \Phi = \emptyset}{?, \Delta, \Pi \vdash M[x, N, (? \setminus \Phi) + y, \Delta][y, P[x, N, ? \setminus \Lambda, \Delta], \Lambda, \Phi] : C} (sub)$$

- Rule d . Without loss of generality we can suppose that $\Psi = ?, \Delta, \Pi$.

$$\frac{\frac{\Pi \vdash M : C \quad \Pi \cap \Delta = \emptyset}{\Delta, \Pi \vdash \Delta M : C \quad ? \cap (\Delta, \Pi) = \emptyset} (weak) \quad (weak)}{?, \Delta, \Pi \vdash ? \Delta M : C}$$

$$\frac{\Pi \vdash M : C \quad (? \cup \Delta) \cap \Pi = \emptyset}{?, \Delta, \Pi \vdash (? \cup \Delta)M : C} (weak)$$

(b_1)	$(\lambda x : A.M)N$	$\rightarrow M[x, N, \emptyset, \emptyset]$	
(b_2)	$(\Delta(\lambda x : A.M))N$	$\rightarrow M[x, N, \emptyset, \Delta]$	
(f)	$(\lambda y : A.M)[x, N, ?, \Delta]$	$\rightarrow \lambda y : A.M[x, N, ? + y, \Delta]$	if $y \notin FV(N)$
(a)	$(MP)[x, N, ?, \Delta]$	$\rightarrow (M[x, P, ?, \Delta]P[x, N, ?, \Delta])$	
(e_1)	$\Lambda M[x, N, ?, \Delta]$	$\rightarrow (\Delta \cup (\Lambda \setminus x))M$	$x \in \Lambda$
(e_2)	$\Lambda M[x, N, ?, \Delta]$	$\rightarrow (? \cap \Lambda)M[x, N, ? \setminus \Lambda, \Delta \cup (\Lambda \setminus ?)]$	$x \notin \Lambda$
(n_1)	$y[x, N, ?, \Delta]$	$\rightarrow y$	$y \neq x$
(n_2)	$x[x, N, ?, \Delta]$	$\rightarrow ?N$	
(c_1)	$M[y, P, \Lambda, \Phi][x, N, ?, \Delta]$	$\rightarrow M[y, P[x, N, ? \setminus \Lambda, \Delta], \Lambda, \Delta \cup (\Phi \setminus x)]$	$x \in \Phi \setminus \Lambda$
(c_2)	$M[y, P, \Lambda, \Phi][x, N, ?, \Delta]$	$\rightarrow M[x, N, (? \setminus \Phi) + y, \Delta]$ $[y, P[x, N, ? \setminus \Lambda, \Delta], \Lambda, ? \cap \Phi]$	$x \notin \Phi \cup \Lambda$
(d)	$? \Delta M$	$\rightarrow (? \cup \Delta)M$	

Figure 4: Reduction Rules of the λ_l -calculus with named variables

We define the reduction rules only on typed terms, since we are focusing here on a named version of the *typed* λ_l calculus with indices. These rules already give the flavor of what a general notion of reduction for non-typed terms with names should be, but a precise formalization of the untyped case is left for further work.

The reduction rules of the typed λ_l -calculus with names are given in Figure 4 (notice that rule b_1 is a particular case of rule b_2 with $\Delta = \emptyset$).

As customary in explicit substitutions calculi with names [3], we work modulo α -conversion, so that we can suppose that in the rule *Weak* the set Δ does not contain variables that are bound in M . Also, this allows us to restrict rule f , without loss of generality, to the case where no variable capture arise.

6 Strong normalization of the λ_l calculus with names

We now give the translation of the terms of λ_l with names into proof nets in PN , and the proof of strong normalization of λ_l .

6.0.1 Translation and simulation of λ_l with names

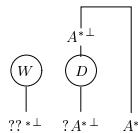
In order to translate a term of λ_l into a proof net, we use exactly the same translation of types that we used in Section 4.2 and we then define the translation of a term M using the type derivation of M .

The translation is then defined in the following way:

- If the term is a variable having a type derivation ending with

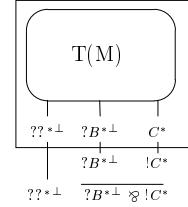
$$\overline{?, x : A \vdash x : A} \text{ Axiome}$$

its translation is given by



- If the term is a λ -abstraction having a type derivation ending with

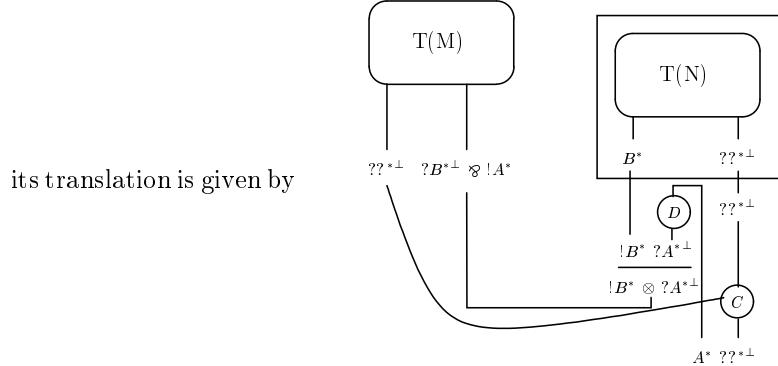
$$\frac{x : B, ? \vdash M : C}{? \vdash \lambda x : B. M : B \rightarrow C} \text{ Lambda}$$



its translation is given by

- If the term is an application having a type derivation ending with

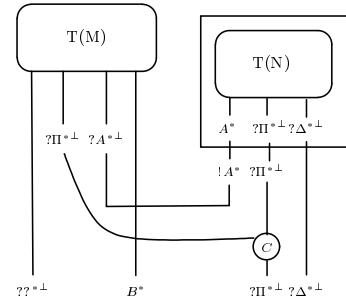
$$\frac{? \vdash M : B \rightarrow A \quad ? \vdash N : B}{? \vdash (MN) : A} \text{ App}$$



- If the term is a substitution having a type derivation ending with

$$\frac{\Delta, \Pi \vdash N : A \quad ?, x : A, \Pi \vdash M : B \quad (? , x : A) \cap \Delta = \emptyset}{\Delta, ?, \Pi \vdash M[x, N, ?, \Delta] : B} \text{ Subst}$$

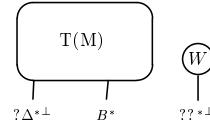
its translation is given by



- If the term is a label having a type derivation ending with

$$\frac{? \vdash M : A \quad ? \cap \Delta = \emptyset}{?, \Delta \vdash \Delta M : A} \text{ Weak}$$

its translation is given by



We can clearly verify that the translation is identical to that given for λ_l with de Bruijn indices. This is completely normal since the type derivations are similar in both formalisms.

The simulation of the reduction rules of the λ_l -calculus with names by the reduction R_E is identical to that given in Section 4.2 for the λ_l -calculus with indices. We just remark that rule n_3 has no sense in the formalism with names so that the proof has one less case. We just state the result without repeating a boring verification:

Lemma 6.1 (Simulation of λ_l with names) If t λ_l -reduces to t' in the formalism with names, then $T(t) \xrightarrow{+_{R_E}} T(t')$, except for the rules e_2 and d for which we have $T(t) = T(t')$.

We can then conclude the following:

Theorem 6.2 (Strong Normalization of λ_l with names) The typed λ_l -calculus with names is strongly normalizing.

7 Conclusion and future works

In this paper we enriched the standard notion of cut elimination in proof nets in order to obtain a system R_E which is flexible enough to provide an interpretation of λ -calculi with explicit substitutions and which is much simpler than the one proposed in [10]. We have proved that this system is strongly normalizing.

We have then proposed a natural translation from λ_l into proof nets that immediately provides strong normalization of the typed version of λ_l , a calculus featuring full composition of substitutions. The proof is extremely simple w.r.t the proof of PSN of λ_l given in [8] and shows in some sense that λ_l , which was designed independently of proof nets, is really tightly related to reduction in proof nets.

Finally, the fact that the relative order of variables is lost in the proof-net representation of a term lead us to discover a version of typed λ_l with named variables, instead of de Bruijn indices. This typed named version of λ_l gives a better understanding of the mechanisms of the calculus. In particular, names allow to understand the manipulation of explicit weakenings in λ_l without entering into the details of renaming of de Bruijn indices. However, the definition of a general notion of reduction for non-typed terms with names remains as further work.

This work suggests several interesting directions for future investigation: on the linear logic side, one should wonder whether R_E is the definitive system able to interpret β reduction, or whether we need some more equivalences to be added. Indeed, there are still a few cases in which the details of a sequent calculus derivation are inessential, even if we did not need to consider them for the purpose of our work, like for example

$$\frac{\vdash ?, B}{\vdash ?A, ?, B} \text{ Weakening} \quad \frac{\vdash ?, B}{\vdash ?, !B} \text{ Box}$$

$$\frac{}{\vdash ?A, ?, !B} \text{ Box} \quad \frac{\vdash ?, !B}{\vdash ?A, ?, !B} \text{ Weakening}$$

On the explicit substitutions side, we look forward to the discovery of a calculus with multiple substitutions with the same properties as λ_l , in the spirit of λ_σ .

Acknowledgments

We would like to thank Bruno Guillaume and Pierre-Louis Curien for their interesting remarks.

References

- [1] M. Abadi, L. Cardelli, P. L. Curien, and J.-J. Lévy. Explicit substitutions. *Journal of Functional Programming*, 4(1):375–416, 1991.
- [2] S. Abramsky and R. Jagadeesan. New foundations for the geometry of interaction. In *Proc. of LICS*, pages 211–222, 1992.
- [3] R. Bloo. *Preservation of Termination for Explicit Substitution*. PhD thesis, Eindhoven University of Technology, 1997.
- [4] R. Bloo and K. Rose. Preservation of strong normalization in named lambda calculi with explicit substitution and garbage collection. In *Computing Science in the Netherlands*, pages 62–72. Netherlands Computer Science Research Foundation, 1995.
- [5] V. Danos. *La logique linéaire appliquée à l'étude de divers processus de normalisation (et principalement du λ -calcul)*. PhD thesis, Université de Paris VII, 1990. Thèse de doctorat de mathématiques.
- [6] V. Danos, J.-B. Joinet, and H. Schellinx. Sequent calculi for second order logic. In J.-Y. Girard, Y. Lafont, and L. Regnier, editors, *Advances in Linear Logic*. Cambridge University Press, 1995.
- [7] V. Danos and L. Regnier. Proof-nets and the Hilbert space. In J.-Y. Girard, Y. Lafont, and L. Regnier, editors, *Advances in Linear Logic*, pages 307–328. Cambridge University Press, London Mathematical Society Lecture Notes, 1995.
- [8] R. David and B. Guillaume. The λ_l -calculus. In *Proceedings of WESTAPP*, pages 2–13, Trento, Italy, 1999.
- [9] R. Di Cosmo and S. Guerrini. Strong normalization of proof nets modulo structural congruences. In P. Narendran and M. Rusinowitch, editors, *Proc of RTA*, volume 1631 of *LNCS*, pages 75–89, Trento, Italy, 1999. Springer Verlag.
- [10] R. Di Cosmo and D. Kesner. Strong normalization of explicit substitutions via cut elimination in proof nets. In *Proc of LICS*, pages 35–46, Warsaw, Poland, 1997.
- [11] M. C. Ferreira, D. Kesner, and L. Puel. Lambda-calculi with explicit substitutions preserving strong normalization. *Applicable Algebra in Engineering Communication and Computing*, 9(4):333–371, 1999.
- [12] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–101, 1987.
- [13] J.-Y. Girard. Geometry of interaction I: interpretation of system F. In R. Ferro, C. Bonotto, S. Valentini, and A. Zanardo, editors, *Logic colloquium 1988*, pages 221–260. North Holland, 1989.
- [14] G. Gonthier, M. Abadi, and J.-J. Lévy. The geometry of optimal lambda reduction. In *Proc. of POPL*, pages 15–26, Albuquerque, New Mexico, 1992. ACM Press.
- [15] B. Guillaume. *Un calcul de substitution avec Étiquettes*. PhD thesis, Université de Savoie, 1999.
- [16] J. Lamping. An algorithm for optimal lambda calculus reduction. In *Proc. of POPL*, pages 16–30, San Francisco, California, 1990. ACM Press.
- [17] P.-A. Mellies. Typed λ -calculi with explicit substitutions may not terminate. In M. Dezani-Ciancaglini and G. Plotkin, editors, *Proc of TLCA*, volume 902 of *LNCS*, April 1995.
- [18] K. Rose. Explicit cyclic substitutions. In Rusinowitch and Rémy, editors, *Proc. of CTRS*, number 656 in *LNCS*, pages 36–50, 1992.