On Proof Normalization in Linear Logic*

Didier Galmiche and Guy Perrier

CRIN - CNRS & INRIA Lorraine
Campus Scientifique - B.P. 239
54506 Vandœuvre-les-Nancy Cedex
France
e-mail: galmiche{perrier}@loria.fr

Abstract

We present a proof-theoretic foundation for automated deduction in linear logic. At first, we systematically study the permutability properties of the inference rules in this logical framework and exploit these to introduce an appropriate notion of forward and backward movement of an inference in a proof. Then we discuss the naturally-arising question of the redundancy reduction and investigate the possibilities of proof normalization which depend on the proof search strategy and the fragment we consider. Thus, we can define the concept of normal proof that might be the basis of works about automatic proof construction and design of logic programming languages based on linear logic.

1 Introduction

Linear logic is a powerful and expressive logic with connections to a variety of topics in computer science. We are mainly interested by the significance it may have in different domains as logic programming or program synthesis through theorem proving. As a matter of fact, classical linear logic (denoted CLL) is a logic of actions introducing notions like controlled and strict resource management [12, 13]. It disallows both weakening and contraction in general although they are introduced for local use through modalities and it conserves a constructive character with a deep symmetry. Linear logic can be an appropriate framework to study logic programming (better than intuitionistic or classical logic) [3, 6, 16], concurrent aspects in logic programming [4, 18], Petri nets reachability [24]. The main point is the resource-sensitive aspect of this logic used, for example, for functional programming [19]. In previous works, we considered the synthesis of correct programs using a theorem proving approach in constructive logics with extraction of programs from proofs [8, 9]. Based on intuitionistic logics , they can present some limits due to the non-symmetrical character of such logics. But we could try to consider this approach in linear logic through an appropriate λ-calculus as logical language [1, 20].
In fact, the first point is to understand what a proof or a proof net is, in some fragments of linear logic, and also to be able to construct proofs of linear logic formulas. Hence, a theorem prover in linear logic can be a first step towards some effective applications of CLL based on the development of proofs like logic programming. For efficiency reasons, the construction of a linear logic prover imposes of course restrictions to an adequate fragment of CLL. Some recent works have been devoted to this important topic [14, 16]. Starting from the fragment of linear

*Accepted for publication in Theoretical Computer Science, vol. 135, december 1994

Horn clauses, they try to extend it in various following ways. The difficulty is then to extend the expressiveness of the language and to keep, at the same time, the efficiency of the initial kernel. Concerning efficiency criteria, we noticed that these works had a common point: the necessity to normalize, as much as possible, the proofs in these fragments, using some properties about inference permutability. Aiming also to extend logic programming, [2] has chosen another approach that consists in, starting from full linear logic, searching for equivalence between proofs in order to normalize them with the central notions of invertibility and focusing. In [27], we have a study about proof search strategies in CLL for a bottom-up direction with the same notions and also for a top-down direction based on specific resolution method. Even if, it is not the central point, permutability properties appears in these approaches. It is a classical concept in works on the conception of efficient proof search methods in non-classical logics [26, 29]. Thus, our approach begins with a systematical study of the inference permutability possibilities in full linear logic aiming to efficient proof construction mechanization. A first attempt in this direction has been developed in [10] where we have focused on the additive and multiplicative fragment of CLL and proposed an algorithm for automated deduction in this fragment.
In this paper we completely refine the permutability notion and we extend the approach to full linear logic. Thus, after having systematically studied, the inference permutability properties, we define specific movements of inference in a proof and we analyze the redundancy reduction in a proof. Then, we are able to propose new proof forms (called normal proofs) that define complete and tractable proof subclasses and we discuss its interest for proof construction. Compared to other approaches on bottom-up and top-down theorem proving, we mainly focus on logical bases through a complete study of the inference permutabilities that could justify further choices of some fragments of CLL adequate for applications as logic programming. From this point, we should be in a position to study more seriously, in further work, the linear logic as a good framework for automated deduction, logic programming and also for the programming with proofs approach.
We present in section 2 the linear logic framework (language and inference system) and recall some basic definitions. Section 3 shows a complete study of the permutability properties of inferences and, in section 4, we deal with the notion of movement of an inference in a proof. Section 5 presents how to reduce some redundancies in a proof by appropriate reduction and by cut elimination. Section 6 defines the notion of normal proof and to use it for the mechanization of proof construction in CLL in an adapted interpretor. In section 7, we emphasize the importance of this approach for designing logic programming languages in adequate fragments of linear logic. Finally, in section 8, we discuss the connections with related works in different fragments of CLL and then we conclude on the usefulness of these results for application embedding linear logic proofs development.

2 Linear logic

Classical linear logic (CLL) has been introduced by Girard [12] as a logic of actions. Born from the semantics of second order lambda-calculus, linear logic is more expressive than traditional logics (classical or intuitionistic ones). Compared to classical logic, two structural rules *weakening* and *contraction* are dropped from the Gentzen-type rules and thus we obtain a system where each resource (hypothesis) must be used exactly once. Thus, conjunction and disjunction, that allow resource sharing, are split into a *multiplicative* version ($\otimes, \wp$) disallowing resource sharing and an *additive* version ($\&, \oplus$) requiring resource sharing. To restore power of classical logic two modal operators ! and ? are introduced knowing that $!F$ allows unlimited consumption of

F and $?F$ allows unlimited use of F. Moreover, the logical constants *true, false* are split into four constants $\mathbf{1}$, $\perp$, $\mathbf{0}$ and $\top$ and an involutive *negation* (denoted $(.)^{\perp}$) is introduced.
Characterized by the absence of structural rules and by a specific treatment of the negation, CLL has proofs that can be considered as actions and introduces a dynamical resource management in these proofs without directional character (no distinction between input and output). We refer the reader to [12, 13, 25, 28] for a broad explanation of the purpose and the meaning of linear logic.

2.1 The language

The language consists of
a) a set of finite *terms* $Term[V]$ on a countable set of variables V,
b) a countable set of *atoms* At each having an arity.
It allows to construct a set $Atom$ of *atomic formulas*: if n is the arity of the atom a and $t_1, t_2, ..., t_n \in Term[V]$ then $a(t_1, t_2, ..., t_n)$ is an atomic formula,
c) a set of *logical operators* $Op = \{\mathbf{0}, \mathbf{1}, \perp, \top, ()^{\perp}, !, ?, \otimes, \wp, \&, \oplus, \forall, \exists\}$. It allows to construct a set $Form$ of *formulas* that are the formulas of CLL, following the grammar

$$F ::= \mathbf{0} | \mathbf{1} | \perp | \top | A | F^{\perp} | ?F | !F | F \otimes F | F \wp F | F \& F | F \oplus F | \forall x F | \exists x F.$$

with $A \in Atom$ and $x \in V$.

Here we manipulate sequents without left-hand sides, using the following notation conventions: variables $\in V$ will be referred by the letters u, x, y, z, terms $\in Term[V]$ by the letters r, s, t, atoms $\in At$ by the letters a, b, c, atomic formulas $\in Atom$ by the letters A, B, C, formulas $\in Form$ by the letters F, G, H, multisets of formulas $\in Form$ by the letters Γ, Δ. Moreover, the letters can possibly be indexed by integers.

2.2 The linear sequent calculus

The inference system we use is the classical linear sequent calculus [12]. Let us recall that a sequent $\vdash F_1, ..., F_n$ is a finite multiset of formulas of $Form$. Thus, we implicitly take into account the exchange rule and consider the commutative linear logic. We present now the inference rules of the linear sequent calculus.

1) Identity Group

$$\frac{}{\vdash A, A^{\perp}}\ ax \qquad\qquad \frac{\vdash F, \Gamma \quad \vdash F^{\perp}, \Delta}{\vdash \Gamma, \Delta}\ Cut$$

2) Structural Group

$$\frac{\vdash \Delta}{\vdash ?F, \Delta}\ w? \qquad\qquad \frac{\vdash ?F, ?F, \Delta}{\vdash ?F, \Delta}\ c?$$

3) Logical Group

$\circ$ *Multiplicative rules*

$$\frac{\vdash F_1, \Gamma_1 \quad \vdash F_2, \Gamma_2}{\vdash F_1 \otimes F_2, \Gamma_1, \Gamma_2}\ \otimes \qquad\qquad \frac{\vdash F_1, F_2, \Gamma}{\vdash F_1 \wp F_2, \Gamma}\ \wp$$

$$\frac{}{\vdash \mathbf{1}}\ \mathbf{1} \qquad \frac{\vdash \Gamma}{\vdash \bot, \Gamma}\ \bot$$

∘ *Additive rules*

$$\frac{\vdash F_1, \Gamma \quad \vdash F_2, \Gamma}{\vdash F_1 \& F_2, \Gamma}\ \& \qquad \frac{\vdash F_1, \Gamma}{\vdash F_1 \oplus F_2, \Gamma}\ \oplus_1 \qquad \frac{\vdash F_2, \Gamma}{\vdash F_1 \oplus F_2, \Gamma}\ \oplus_2 \qquad \frac{}{\vdash \top, \Delta}\ \top$$

∘ *Exponential rules*

$$\frac{\vdash F, ?\Gamma}{\vdash !F, ?\Gamma}\ ! \qquad \frac{\vdash F, \Gamma}{\vdash ?F, \Gamma}\ ?$$

∘ *Quantifiers rules*

$$\frac{\vdash F[y/x], \Gamma}{\vdash \forall x F, \Gamma}\ \forall \qquad \frac{\vdash F[t/x], \Gamma}{\vdash \exists x F, \Gamma}\ \exists$$

In the $\forall$ rule, y is not free in Γ and in F if y is different of x.
Let us remark that there is no rule for the constant $\mathbf{0}$ and that the *linear negation*, that is essential for the symmetrical character of CLL, is defined by the following equalities:

$$\begin{array}{l}
F^{\perp\perp} = F,\ \mathbf{1}^{\perp} = \bot,\ \bot^{\perp} = \mathbf{1},\ \top^{\perp} = \mathbf{0},\ \mathbf{0}^{\perp} = \top, \\
(F \otimes G)^{\perp} = F^{\perp} \wp\, G^{\perp} \text{ and } (F \wp\, G)^{\perp} = F^{\perp} \otimes G^{\perp}. \\
(F \& G)^{\perp} = F^{\perp} \oplus G^{\perp} \text{ and } (F \oplus G)^{\perp} = F^{\perp} \& G^{\perp}. \\
(\forall x F)^{\perp} = \exists x F^{\perp} \text{ and } (\exists x F)^{\perp} = \forall x F^{\perp}. \\
(!F)^{\perp} = ?F^{\perp} \text{ and } (?F)^{\perp} = !F^{\perp}.
\end{array}$$

Remark 2.1 *If we consider full linear logic, it is equivalent to work with sequents without left-hand side part (system called CLL) or with left-hand side part (system called CLL'). It is due to the property of the linear negation that allows to move a formula from one side to the other in a sequent, by changing it into its negation. We have chosen to work in CLL for simplification purposes but, for applications as logic programming, CLL' presents some advantages: for example, it can be restricted without difficulty to a fragment of linear logic without negation. As a matter of fact, the translation of the results for CLL in CLL' can be easily performed.*

2.3 Derivations in CLL

Before proceeding further in our study, it is necessary to fix some vocabulary relatively to the notion of derivation and proof in CLL. Let us recall that an *inference* is an instance of a rule of the system defined above and the *type* of an inference I is the name of the corresponding rule, that is denoted by $type(I)$
In a derivation, it will be necessary to follow the evolution of the formulas from the time when they are introduced (called *principal*) to the time when they are used, disappearing or becoming subformulas (called *active*). To take into account this evolution, it will be necessary to mark them in the derivations. We thus give the following definitions.

2.3.1 Principal and active formulas, contexts

Definition 2.1 *A* principal formula *of an inference I, such that $type(I) \neq c?$, is a formula of the conclusion that did not exist in the premises.*

If type(I) = c? the principal formula is the result of the contraction of the two formulas in the premise.

Definition 2.2 *The* principal part *of an inference I (denoted by $\Delta_p(I), \Delta'_p(I), ...$) is the multiset of its principal formulas.*

Example 2.1 *For the inference* $I_1 \equiv \dfrac{}{\vdash A, A^\perp}$ *we have* $\Delta_p(I_1) = \{A, A^\perp\}$.

For the inference $I_2 \equiv \dfrac{\vdash F, \Delta \qquad \vdash F^\perp, \Delta'}{\vdash \Delta, \Delta'}$ *we have* $\Delta_p(I_2) = \emptyset$.

Definition 2.3 *An* active formula *of an inference I, such that type(I) $\neq$ c?, is a formula in a premise that does not exist in the conclusion.*
If type(I) = c?, I has two active formulas (?F,?F) that are the ones contracted in one formula (?F).

Definition 2.4 *The* active part *of the i-th premise of an inference I (denoted by $\Delta^i_a(I)$) is the multiset of its active formulas.*

Example 2.2 *Considering* $I \equiv \dfrac{\vdash F, G, \Gamma}{\vdash F \wp G, \Gamma}$ *we have* $\Delta^1_a(I) = \{F, G\}$.

For $I \equiv \dfrac{\vdash \Gamma}{\vdash ?F, \Gamma}$ *we have* $\Delta^1_a(I) = \emptyset$.

Considering the inference $I \equiv \dfrac{\vdash \Gamma_1, F \qquad \vdash \Gamma_2, G}{\vdash \Gamma_1, \Gamma_2, F \otimes G}$ *we have* $\Delta^1_a(I) = \{F\}$ *and* $\Delta^2_a(I) = \{G\}$.

Definition 2.5 *The* context *of an inference premise is the complement of its active part.*

We can classify the inference rules in two categories: the ones depending on the context of the premises (those of the &, ! or $\forall$ type) and the others. It is due to the fact that the application of the rules &, ! or $\forall$ is possible under some conditions on the premise contexts. For &, the contexts of the two premises have to be identical, for ! the context of the premise has to be of the form $?\Delta$ and for $\forall$ the variable y associated to the inference cannot be free in the premise context.

2.3.2 Marked derivations and proofs

We use the classical representation with binary trees labelled with sequents for defining the notions of derivation (or deduction) and proof [7]. Let us recall some definitions.

Definition 2.6 *(i) An* hypothesis *of a derivation is a sequent labelling one of its leaves.*
(ii) An intermediate conclusion *is a sequent that is not an hypothesis (i.e, not labelling a leave).*
(iii) The conclusion *of a derivation is the sequent labelling the root of this tree.*
(iv) A proof *in CLL is a derivation without hypotheses.*
(v) A sequent $\vdash \Delta$ is provable *if there exists a proof with $\vdash \Delta$ as conclusion.*

It appears important to be able to follow the evolution of a formula in a proof and to do that we consider the *marking* of each of its inferences.

Definition 2.7 *A* marked inference *is an inference with a function from the premises to the conclusion, that allows to identify formulas in the premises with formulas or sub-formulas in the conclusion. A* marked proof *is a proof where each inference is marked.*

We can illustrate this notion with the following example

Example 2.3 *In the case of* $\vdash A, ?A \oplus F, ?A$*, we can obtain two different marked proofs*

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash A, A_1^\perp}}{\vdash A, ?A_1^\perp}}{\vdash A, ?A_1^\perp, ?A_2^\perp}}{\vdash A, ?A_1^\perp \oplus F, ?A_2^\perp} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\vdash A, A_1^\perp}}{\vdash A, ?A_1^\perp}}{\vdash A, ?A_1^\perp, ?A_2^\perp}}{\vdash A, ?A_1^\perp, ?A_2^\perp \oplus F}$$

In general, the choice is neutral w.r.t. inference properties that we want to study. In the rest of the paper, even it is not explicit, we only consider marked proofs.

Definition 2.8 *Let A be a formula of an intermediate conclusion of a proof Π of CLL, I an inference of Π, we say that A* is introduced by *I if A is a principal formula of I.*

Definition 2.9 *Let A be a formula of an intermediate conclusion of a proof Π of CLL, I an inference of Π, we say that A* is actived in *I if A is an active formula of I.*

Remark 2.2 *A formula can be introduced by several inferences. For example, let us consider the following CLL-proof*

$$\dfrac{\dfrac{\dfrac{}{\vdash A, A^\perp}}{\vdash A \oplus B, A^\perp}\oplus_1 \qquad \dfrac{\dfrac{}{\vdash B, B^\perp}}{\vdash A \oplus B, B^\perp}\oplus_2}{\vdash A \oplus B, A^\perp \& B^\perp}\&$$

The formula $A \oplus B$ of the conclusion is introduced by two inferences: one of type $\oplus_1$ and one of type $\oplus_2$.

Hence, after this presentation of the logical framework we consider, we can now focus on the first important notion that is the permutability of inferences in a proof.

3 Inference permutability

We take as our point of conceptual departure the notion of inference permutability in full linear logic. It is a basic notion that appears to be very important for an efficient proof search (and theorem proving) in non-classical logics [26, 29].
We want to systematically study the possibilities we can obtain, from a given proof in CLL, other proofs by a simple permutation of two inferences that are consecutive in the proof tree. As an example, we take a proof where an inference of type & immediately precedes an inference of type $\oplus_1$. Let us consider the proof Π:

$$\dfrac{\dfrac{\Pi_1\left\{\begin{array}{c}\vdots\\ \vdash F, G, \Delta\end{array}\right. \qquad \Pi_2\left\{\begin{array}{c}\vdots\\ \vdash F, H_1, \Delta\end{array}\right.}{\vdash F, G\&H_1, \Delta}\&}{\vdash F \oplus H_2, G\&H_1, \Delta}\oplus_1$$

From this form, we can easily deduce the following proof Π'

$$\dfrac{\dfrac{\Pi_1\left\{\begin{array}{c}\vdots\\ \vdash F, G, \Delta\end{array}\right.}{\vdash F \oplus H_2, G, \Delta}\oplus_1 \qquad \dfrac{\Pi_2\left\{\begin{array}{c}\vdots\\ \vdash F, H_1, \Delta\end{array}\right.}{\vdash F \oplus H_2, H_1, \Delta}\oplus_1}{\vdash F \oplus H_2, G\&H_1, \Delta}\&$$

This illustrates that the permutation between an inference of type & and an inference of type $\oplus_1$ is always possible in one direction. The following counter-example shows that it is not possible in the other direction. Let us consider the proof

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash A, A^\perp}id \quad \dfrac{}{\vdash B, B^\perp}id}{\vdash A \otimes B, A^\perp, B^\perp}\otimes}{\vdash (A \otimes B) \oplus (C \otimes B), A^\perp, B^\perp}\oplus_1 \quad \dfrac{\dfrac{\dfrac{}{\vdash C, C^\perp}id \quad \dfrac{}{\vdash B, B^\perp}id}{\vdash C \otimes B, C^\perp, B^\perp}\otimes}{\vdash (A \otimes B) \oplus (C \otimes B), C^\perp, B^\perp}\oplus_2}{\vdash (A \otimes B) \oplus (C \otimes B), A^\perp \& C^\perp, B^\perp}\&$$

It is clear here that we cannot permutate the inferences of type $\oplus_1$ and &. Thus, it seems interesting to see what happens for each pair of inference types and then to analyze the consequences of the permutability opportunities. Let us remark that, when two inferences are consecutive in a proof, it is not always relevant to consider their permutations. For example, if a principal formula of the first inference is active in the second inference then the order cannot be changed. Hence, the following appropriate definition

Definition 3.1 *Let us consider a proof* Π *in CLL, we say that* I_1 *and* I_2 *two inferences of* Π *are* in permutation position *if they verify the conditions:*
(α) I_2 *follows directly* I_1 *in* Π *(denoted by* $I_1 \backslash_\Pi I_2$*)* ,
(β) the principal part in I_1 *is disjoint of the active part of j-th premise of* I_2 *where it appears, i.e.,* $\Delta_p(I_1) \neq \Delta_a^j(I_2)$.

Intuitively, the notion of permutability is easy to understand. It means the possibility to invert two inferences in a proof without disturbing the rest of the proof (the parts below and above the inferences). In a previous work on deduction in the additive and multiplicative fragment of CLL [10], we have called it perfect-permutability (*pp*). Should one inference be of type &, there is some difficulty because of the duplication of the other inference (due to the duplication in the context). In [10] we have translated this variant by the notion of quasi-permutability (*qp*). But such definitions were rough and here we have refined them and embedded it in the following general definition.

Definition 3.2 *Let us consider a proof* Π *in CLL,* I_1 *and* I_2 *inferences of* Π *being in permutation position,* I_1 is permutable with I_2 in Π *if there exists inferences* I_1' *and* I_2' *such that*
(i) type(I_1') = *type*(I_1) *and type*(I_2') = *type*(I_2),
(ii) the conclusion of I_2' *coincides with a premise of* I_1',
(iii) if type(I_2) = & *and* J_1 *is the other inference immediately preceding* I_2 *in* Π *then type*(J_1) = *type*(I_1).
(iv) if type(I_1)= & *there exists an other inference* J_2', *such that type*(J_2') = *type*(I_2) *and the*

conclusion of which coincides with the second premise of I'_1.
(v) Let us consider the derivation (called permutation object*) composed by* I_1 *(and* J_1 *if type(*I_2*) =* &*) followed by* I_2 *and the derivation (called* permutation result*) composed by* I'_2 *(and* J'_2 *if type(*I_1*)=* &*) followed by* I'_1*, both have the same conclusion and the same hypotheses modulo a duplication of some of them and a renaming of certain free variables.*
In the other cases, we say that I_1 *is not permutable with* I_2.

Let us illustrate the definition with the example and the counter-example presented in the beginning of this section.

Example 3.1 *Let us consider the previous proof, named* Π, *with*

$$I_1 = \frac{\vdash F, G, \Delta \quad \vdash F, H_1, \Delta}{\vdash F, G\&H_1, \Delta}\ \& \quad \textit{and} \quad I_2 = \frac{\vdash F, G\&H_1, \Delta}{\vdash F \oplus H_2, G\&H_1, \Delta}\ \oplus_1$$

I_1 *and* I_2 *are in permutation position because conditions* α *and* β *(Definition 3.1) are satisfied, i.e.,* $\Delta_p(I_1) = \{G\&H_1\}$ *and* $\Delta_a(I_2) = \{F\}$ *are disjoint.*

Let us consider $I'_1 = \dfrac{\vdash F \oplus H_2, G, \Delta \quad \vdash F \oplus H_2, H_1, \Delta}{\vdash F \oplus H_2, G\&H_1, \Delta}\ \&$ *and* $I'_2 = \dfrac{\vdash F, G, \Delta}{\vdash F \oplus H_2, G, \Delta}\ \oplus_1$,

the conditions (i), (ii), (iii) (Definition 3.2) are satisfied.
Moreover, I_1 *being of type* &, *there exists* $J'_2 = \dfrac{\vdash F, H, \Delta}{\vdash F \oplus H_2, H, \Delta}\ \oplus_1$ *that verifies the condition (iv).*
The condition (v) is also easily satisfied and then I_1 *is permutable with* I_2.

Example 3.2 *Let us consider the previous proof denoted* Π' *with*

$$I_1 = \frac{\vdash A \otimes B, A^\perp, B^\perp}{\vdash (A \otimes B) \oplus (C \otimes B), A^\perp, B^\perp}\ \oplus_1$$

$$I_2 = \frac{\vdash (A \otimes B) \oplus (C \otimes B), A^\perp, B^\perp \quad \vdash (A \otimes B) \oplus (C \otimes B), C^\perp, B^\perp}{\vdash (A \otimes B) \oplus (C \otimes B), A^\perp\&C^\perp, B^\perp}\ \&$$

I_1 *and* I_2 *are in permutation position because conditions* α *and* β *(Definition 3.1) are verified, i.e.,*
$\Delta_p(I_1) = \{(A \otimes B) \oplus (C \otimes D)\}$ *and* $\Delta_a(I_2) = \{A^\perp\}$ *are disjoint.*
But the condition (iii) is not verified and thus I_1 *is not permutable with* I_2.

Definition 3.3 *A proof* Π' is obtained by permutation *of* I_1 *with* I_2 *in* Π *if it is obtained by replacing the* permutation object *relatively to* I_1 *and* I_2 *by the* permutation result *(modulo a renaming of free variables and duplication of tree branchs above the permutation object)*

The operation of permutation of two inferences in a proof implies (because of its involutive character) an equivalence relation between proofs.

Definition 3.4 *Let* Π *and* Π' *be two proofs of CLL,* Π' *is equivalent to* Π *modulo an inference permutation (denoted by* $\Pi' \sim \Pi$*) if there exists two inferences* I_1 *and* I_2 *in* Π *such that* Π' *is obtained by permutation of* I_1 *with* I_2 *in* Π.

Definition 3.5 *Let Π and Π' two proofs of CLL, Π' is equivalent to Π modulo the inference order (denoted $\Pi' \sim^* \Pi$) if there exists a finite sequence $\Pi_1, ..., \Pi_n$ of n proofs in CLL ($n \geq 1$) such that (i) $\Pi_1 \equiv \Pi$ and $\Pi_n \equiv \Pi'$, (ii) for $i \in [1, n-1], \Pi_i \sim \Pi_{i+1}$.*

Theorem 3.1 *The relation $\sim^*$ is an equivalence relation*

We will now systematically study the permutability properties of two inferences in a proof according to their types. Let us remark that the types $\oplus_1$ and $\oplus_2$ have the same properties and then we gather them into one denoted $\oplus$.

Theorem 3.2 *(permutability theorem)*
Let t_1 and t_2 be two types of inference, in the following array,
(i) the case(t_1, t_2) in the following array contains p if and only if for any inferences I_1 and I_2 of type t_1 and t_2 being in permutation position in a proof Π, I_1 is permutable with I_2.
(ii) the case(t_1, t_2) contains np if and only if there exists two inferences I_1 and I_2 of type t_1 and t_2 being in permutation position in a proof Π, which are not permutable.
(iii) the case(t_1, t_2) contains a cross $\times$ if and only if for any inferences I_1 and I_2 of type t_1 and t_2 in a proof Π, I_1 is never in permutation position with I_2.

$\mathbf{t_2 \backslash t_1}$	cut	$\otimes$	$\wp$	&	$\oplus$	?	w?	c?	!	$\forall$	$\exists$	$\perp$
cut	p	p	p	p	p	p	p	p	np	p	p	p
$\otimes$	p	p	p	p	p	p	p	p	np	p	p	p
$\wp$	np	np	p	p	p	p	p	p	np	p	p	p
&	np	np	np^*	np^*	np	np	np	np^*	np	np^*	np	np^*
$\oplus$	p	p	p	p	p	p	p	p	np	p	p	p
?	p	p	p	p	p	p	p	p	p	p	p	p
$w?$	p	p	p	p	p	p	p	p	p	p	p	p
$c?$	np	np	p	p	p	p	p	p	p	p	p	p
!	np	$\times$	$\times$	$\times$	$\times$	np	p	p	$\times$	$\times$	$\times$	$\times$
$\forall$	np	p	p	p	p	p	p	p	np	p	np	p
$\exists$	p	p	p	p	p	p	p	p	np	p	p	p
$\perp$	p	p	p	p	p	p	p	p	np	p	p	p

Proof 3.1 *By case analysis according to the partition of the inference rules into two groups: the ones that depend on the context, i.e, of type &, !, $\forall$ and the other ones. The complete proof is given in the appendix A.*

Let us remark that the np^* in the line of & indicate that this non-permutability is relative to our definition but it can be overcome by a special treatment (see subsection 4.2).
We can analyze the array presented in theorem 3.2, column by column. When a column, like the one of $\wp$, contains only one np, it means that we will be able to move forward (or down) an inference of type $\wp$ in a proof. By repetition of this elementary operation, we will move forward, as far as possible, such an inference. But two problems can stop such a movement: an inference of type & or an inference where the principal formula of the other one becomes active (the inferences are no more in permutation position). We can also analyze the array line by line. When a line, like the one of ? contains only p, it means that we will be able to move backward (or up) an inference of type ? in a proof. By repetition of this elementary operation, we will move backward, as far as possible, such an inference. The only problem that can stop

such a movement is an inference introducing the active formula of the other (the inferences are no more in permutation position).
This double analysis leads us to classify the inference types into two groups: the ones we can move backward and the others we can move forward as far as possible, in a proof. Now, we will study now in details these possible contradictory movements in a proof in full linear logic.

4 Inference movement in a LL proof

In this section, we define the notion of movement of an inference in a proof. We are mainly interested by movements on an inference towards the top of the proof (called *backward or up movement*) and towards the bottom of the proof (called *forward or down movement*).

4.1 Backward movement

Intuitively, it is an iteration of the movement of backward permutation of an inference in a proof. This movement can be complicated by the presence of an inference of type & that duplicates the inference in backward movement. Then, by such a movement, an inference can be duplicated the number of times it goes across inferences of type &. Taking this problem into account leads us to the following definition

Definition 4.1 *Let* Π *be a proof of LL and I an inference of* Π*, a proof* Π' *of LL is* obtained by backward movement of I in Π *if there exists a sequence* $\Pi_0, .., \Pi_n$ *of proofs in LL and a sequence* $Inf_0, ..., Inf_n$ *of inference sets such that*
(i) for any $i \in [0, n]$*,* Inf_i *is an inference set of* Π_i *(inferences open to permutability),*
(ii) $\Pi_0 \equiv \Pi$*,* $\Pi_n \equiv \Pi'$*,* $Inf_0 \equiv \{I\}$*,*
(iii) for any $i \in [0, n-1]$*,* Π_{i+1} *is obtained from* Π_i *by permutation of an inference* I_i^1 *with an inference* I_i^2 *of* Inf_i*.* $I_i'^1$ *and* $I_i'^2$ *(and possibly* $J_i'^2$*) being the corresponding inferences of* Π_{i+1} *then* Inf_{i+1} *is the union of the inferences of* Π_{i+1} *in* Inf_i *and of* $I_i'^2$ *(and possibly* $J_i'^2$*).*
Inf_n *is called the inference set of* Π' *resulting of the backward movement of* I *in* Π*.*

Let us illustrate this definition by an example.

Example 4.1 *Let us consider the proof* Π_0 *of LL*

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash A, A^\perp}}{\vdash A \oplus B, A^\perp}\oplus_1 \qquad \dfrac{\dfrac{}{\vdash B, B^\perp}}{\vdash A \oplus B, B^\perp}\oplus_2}{\vdash A \oplus B, A^\perp \& B^\perp}\& \qquad \dfrac{}{\vdash C, C^\perp}}{\vdash (A \oplus B) \otimes C, A^\perp \& B^\perp, C^\perp}\otimes}{\vdash (A \oplus B) \otimes C, A^\perp \& B^\perp, C^\perp, ?C}w?$$

we can obtained by backward movement of $w?$ *in* Π_0 *the following proof* Π_3

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash A, A^\perp}}{\vdash A, A^\perp, ?C}w?}{\vdash A \oplus B, A^\perp, ?C}\oplus_1 \qquad \dfrac{\dfrac{\dfrac{}{\vdash B, B^\perp}}{\vdash B, B^\perp, ?C}w?}{\vdash A \oplus B, B^\perp, ?C}\oplus_2}{\vdash A \oplus B, A^\perp \& B^\perp, ?C}\& \qquad \dfrac{}{\vdash C, C^\perp}}{\vdash (A \oplus B) \otimes C, A^\perp \& B^\perp, C^\perp, ?C}\otimes$$

We observe that the inference w? get over an inference of type & by backward movement in Π_0*, with its duplication as main effect and then the inference set of* Π_3 *resulting of the movement of w? consists of two inferences w?.*

Let us consider again the array of theorem 3.2, line by line, to see which inference types are well adapted to the backward movement. After a first analysis, it seems to be $cut, \otimes, \oplus, ?, w?, \exists, \perp$, but the last one can also be adequate for forward movement, as we will notice it after.
Concerning the inference of type w?, by the theorem 3.2, there is no problem for backward movement of such inference. Moreover, it does not need an active formula and it is always in permutation position with the inferences that precede it immediately. So we can move it backward up to the axioms. This is illustrated by the above example and expressed by the following theorem

Theorem 4.1 *(backward movement of w?)*
Let Π *be a proof in LL and I an inference of* Π *of the w? type, there exists a proof* Π' *obtained by backward movement of I in* Π *such that any inference I' of* Π' *resulting of this movement is immediately preceded by an axiom.*

Proof 4.1 *By induction on h* ($h \geq 1$)*, height of* Π_1*, subtree of* Π *with I's premise as root.*
- $h = 1$*. I is immediately preceded by an axiom in* Π *and then we can consider* $\Pi' \equiv \Pi$*.*
- *Let us assume the property true for a height h* (≥ 1) *and prove it for* $h+1$*.*

Let us consider a proof Π *and an inference I of* Π *of w? type such that the subproof tree* Π_1*, having the premise of I as root, is of height h+1, the inference* I_1 *that immediately precedes I in* Π*, is not an axiom because* $h+1 \geq 2$*.*
Moreover, I has no active formula and then I_1 *is in permutation position with I and is permutable with it by theorem 3.2.*
1) $type(I_1) \neq \&$
Let Π'' *be the proof obtained by permutation of* I_1 *with I in* Π *and* I'' *the inference of* Π'' *corresponding to I, the subproof tree* Π''_1 *of* Π''*, having* I'' *as root, has the height h and thus we can apply the induction hypothesis to it.*
Then, there exists a proof Π' *obtained by backward movement of* I'' *in* Π'' *such that any inference* I' *of* Π'*, resulting from this movement, is immediately preceded by an axiom.*
By the composition of this backward movement with the permutation that transforms Π *into* Π'*, we obtain a backward movement that leads from* Π *to* Π'*.*
Thus, the property is true for h+1.
2) $type(I_1) = \&$
This case is similar to the previous one except that I is split by the permutation into two inferences I'' *and* J'' *and thus we have to apply the induction hypothesis twice.*

Concerning the type ?, the only difference with the previous one is that each inference of the ? type contains an active formula and its backward movement is stopped by the inferences introducing this active formula. We have the following result

Theorem 4.2 *(backward movement of ?)*
Let Π *be a proof in LL and I an inference of* Π *of the ? type, there exists a proof* Π' *obtained by backward movement of I in* Π *such that any inference I' of* Π' *resulting of this movement is immediately preceded by an inference introducing the active formula of I'.*

Proof 4.2 *The proof is similar to the Theorem 4.1 proof.*

Apparently, by theorem 3.2, the backward movement of a *cut* inference can be hold up by an inference of type !. But, as shown by the example below, this problem can be solved and the *cut* inference has the same behavior than the ? inference and consequently it leads to an analogous result.

Example 4.2 *Let us consider an example of backward movement of a cut inference in the following proof* Π_0

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash B, B^\perp}}{\vdash B, ?B^\perp}\;?}{\vdash B, ?B^\perp, ?A^\perp}\;w?}{\vdash !B, ?B^\perp, ?A^\perp}\;! \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{A, A^\perp}}{\vdash A, ?A^\perp}\;?}{\vdash !A, ?A^\perp}\;! \quad \dfrac{}{\vdash B, B^\perp}}{\vdash !A, (?A^\perp) \otimes B, B^\perp}\;\otimes}{\vdash !B, ?B^\perp, (?A^\perp) \otimes B, B^\perp}\;cut$$

We cannot permute the cut inference with the ! inference on the left-hand side, whereas the permutation is possible with the inference of type $\otimes$.
By such a movement we obtain the proof Π_1

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash B, B^\perp}}{\vdash B, ?B^\perp}\;?}{\vdash B, ?B^\perp, ?A^\perp}\;w?}{\vdash !B, ?B^\perp, ?A^\perp}\;! \qquad \dfrac{\dfrac{\dfrac{}{\vdash A, A^\perp}}{\vdash A, ?A^\perp}\;?}{\vdash !A, ?A^\perp}\;!}{\vdash !B, ?B^\perp, ?A^\perp}\;cut \qquad \dfrac{}{\vdash B, B^\perp}}{\vdash !B, ?B^\perp, (?A^\perp) \otimes B, B^\perp}\;\otimes$$

The permutation of the cut inference with the ! inference in left-hand side is now possible and we obtain the proof Π_2

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash B, B^\perp}}{\vdash B, ?B^\perp}\;?}{\vdash B, ?B^\perp, ?A}\;w? \qquad \dfrac{\dfrac{\dfrac{}{\vdash A, A^\perp}}{\vdash A, ?A^\perp}\;?}{\vdash !A, ?A^\perp}\;!}{\vdash B, ?B^\perp, ?A^\perp}\;cut}{\vdash !B, ?B^\perp, ?A^\perp}\;! \qquad \dfrac{}{\vdash B, B^\perp}}{\vdash !B, ?B^\perp, (?A^\perp) \otimes B, B^\perp}\;\otimes$$

At this time, the movement stops because the cut inference is no longer in permutation position with its immediate predecessors.

Theorem 4.3 *(backward movement of cut)*
Let Π *be a proof in LL and I an inference of* Π *of the cut type, there exists a proof* Π' *obtained by backward movement of I in* Π *such that for any inference I' of* Π' *resulting of this movement, the inferences preceding immediately I' introduce its active formulas.*

Proof 4.3 *By induction on the number* n *of inferences preceding I in* Π.
• $n = 0$*. There is no proof* Π *where I is not preceded by an inference and thus the property is true.*

• *Let us assume the property true until any order n and prove that it is true for n+1.*
Let us consider a proof Π *and I an inference of type cut preceded by n+1 inferences in* Π,
1) One inference immediately preceding I is not of the ! type and does not introduce an active formula of I.
Let I_1 *such an inference, by theorem 3.2, we can permute* I_1 *with I to obtain a proof* Π'' *where the inferences corresponding to I are preceded by n inferences. Thus, we can apply the induction hypothesis and we obtain by backward movement of these inferences in* Π'' *a proof* Π' *verifying the given criteria.*
By composition of these movements with the permutation leading from Π *to* Π'' *we obtain a backward movement from* Π *to* Π'.
2) The inferences immediately preceding I are of type ! or introduce an active formula of I.
2.1) The inferences immediately preceding I introduce the active formulas of I.
In this case the proof Π' *we search is* Π.
2.2) One inference immediately preceding I is of type ! and does not introduce an active formula of I.
Let I_1 *be such an inference being of the form* $\dfrac{\vdash ?F, G, ?\Delta_n}{\vdash ?F, !G, ?\Delta_n}$ *where* $?F$ *is active in I, the other inference preceding immediately I contains* $!F^\perp$ *in this conclusion and necessarily introduces* $!F^\perp$. *Then* Π *has the following form*

$$\dfrac{\dfrac{\vdash ?F, G, ?\Delta_n}{\vdash ?F, !G, ?\Delta_n}\,! \qquad \dfrac{\vdash F^\perp, ?\Delta'_n}{\vdash !F^\perp, ?\Delta'_n}\,!}{\vdash !G, ?\Delta_n, ?\Delta'_n}\,cut$$

We can then permute I_1 *with I to obtain the proof*

$$\dfrac{\dfrac{\vdash ?F, G, ?\Delta_n \qquad \dfrac{\vdash F^\perp, ?\Delta'_n}{\vdash !F^\perp, ?\Delta'_n}\,!}{\vdash G, ?\Delta_n, ?\Delta'_n}\,cut}{\vdash !G, ?\Delta_n, ?\Delta'_n}\,!$$

There are n inferences preceding the new cut and then we can apply the induction hypothesis to them and go on as in the case 1).

Concerning the types $\otimes$, $\oplus$ and $\exists$, the potential obstacle of the inferences of type ! cannot be bypassed as for the cuts. This point is illustrated in the following example.

Example 4.3 *Let us consider a backward movement of* $\otimes$ *in the following proof* Π_0

$$\dfrac{\dfrac{\dfrac{\overline{\vdash A, A^\perp}}{\vdash A\wp A^\perp}\,\wp}{\vdash A\wp A^\perp, ?B}\,w? \qquad \dfrac{\dfrac{\overline{\vdash C, C^\perp}}{\vdash C, ?C^\perp}\,?}{\vdash !C, ?C^\perp}\,!}{\vdash (A\wp A^\perp)\otimes ?C^\perp, !C, ?B}\,\otimes$$

we obtain by permutation of w? with $\otimes$ *the proof* Π_1

$$
\dfrac{\dfrac{\dfrac{}{\vdash A, A^{\perp}}}{\vdash A\wp A^{\perp}}\,\wp \qquad \dfrac{\dfrac{\dfrac{}{\vdash C, C^{\perp}}}{\vdash C, ?C^{\perp}}\,?}{\vdash !C, ?C^{\perp}}\,!}{\dfrac{\vdash (A\wp A^{\perp})\otimes ?C^{\perp}, !C}{\vdash (A\wp A^{\perp})\otimes ?C^{\perp}, !C, ?B}\,w?}\,\otimes
$$

The permutation of $\otimes$ with ! is not possible and the backward movement stops at this step.

Theorem 4.4 *(backward movement of $\otimes$, $\oplus$, $\exists$)*
Let Π be a proof in LL and I an inference of Π of type $\otimes$, $\oplus$ or $\exists$, there exists a proof Π' obtained by backward movement of I in Π, such that any inference I' of Π' resulting of this movement is immediately preceded by an inference that introduces an active formula of I' or by an inference of ! type.

Proof 4.4 *The proof has the same scheme as in the case of weakening.*

It is also possible to move backward (or up) the inferences of ! type in a proof. But, taking into account the particular behaviour of the ! rule, this movement is not possible by successive permutations but only possibly by jumps from an intermediate conclusion of the form $\vdash F, ?\Delta$ to another one of the form $\vdash F, ?\Delta'$, if we do not have & in the fragment we consider.

4.2 Forward movement

Intuitively, it is an iteration of the forward movement of an inference that permutes with another in a proof. This movement can be complicated by the meeting, during this process, of an inference of type & with a contrary effect as for backward movement. Then, by such a movement, the inference is not duplicated but merged with another one that provides from the other branch of the proof tree. Thus, starting from a set of inferences, we terminate the movement with only one inference. That is the reason why the forward movement notion is defined by duality from the backward movement.

Definition 4.2 *Let Π be a proof of LL and I an inference of Π, a proof Π' of LL is obtained by* forward movement of I in Π *if there exists an inference I' of Π' such that Π is obtained by backward movement of I' in Π' and I is one of the resulting inferences.*
I', that is unique, is called the inference resulting of the forward movement *of I in Π.*

Let us illustrate this definition by an example

Example 4.4 *Let us consider a forward movement of $\forall$ in a proof. Let Π_0 be the proof*

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash a(x), a(x)^{\perp}}}{\exists y a(y), a(x)^{\perp}}\,\exists}{\vdash \exists y a(y), \forall z a(z)^{\perp}}\,\forall \qquad \dfrac{}{\vdash B, B^{\perp}}}{\vdash (\exists y a(y))\otimes B, \forall z a(z)^{\perp}, B^{\perp}}\,\otimes}{\vdash (\exists y a(y))\otimes B, \forall z a(z)^{\perp}, (B^{\perp}\oplus c(x))}\,\oplus_1}{\vdash (\exists y a(y))\otimes B, (\forall z a(z)^{\perp})\wp(B^{\perp}\oplus c(x))}\,\wp
$$

By permutation of $\otimes$ *and* $\forall$ *we obtain the proof* Π_1

$$
\begin{prooftree}
\AxiomC{}
\UnaryInfC{$\vdash a(x), a(x)^\perp$}
\RightLabel{$\exists$}
\UnaryInfC{$\vdash \exists y a(y), a(x)^\perp$}
\AxiomC{}
\UnaryInfC{$\vdash B, B^\perp$}
\RightLabel{$\otimes$}
\BinaryInfC{$\vdash (\exists y a(y)) \otimes B, a(x)^\perp, B^\perp$}
\RightLabel{$\forall$}
\UnaryInfC{$\vdash (\exists y a(y)) \otimes B, \forall z a(z)^\perp, B^\perp$}
\RightLabel{$\oplus_1$}
\UnaryInfC{$\vdash (\exists y a(y)) \otimes B, \forall z a(z)^\perp, (B^\perp \oplus c(x))$}
\RightLabel{$\wp$}
\UnaryInfC{$\vdash (\exists y a(y)) \otimes B, (\forall z a(z)^\perp) \wp (B^\perp \oplus c(x))$}
\end{prooftree}
$$

By permutation of $\oplus_1$ *and* $\forall$ *we obtain the proof* Π_2*. The permutation implies the renaming of the variable* x *in the subtree having the conclusion of* $\otimes$ *as root.*

$$
\begin{prooftree}
\AxiomC{}
\UnaryInfC{$\vdash a(u), a(u)^\perp$}
\RightLabel{$\exists$}
\UnaryInfC{$\vdash \exists y a(y), a(u)^\perp$}
\AxiomC{}
\UnaryInfC{$\vdash B, B^\perp$}
\RightLabel{$\otimes$}
\BinaryInfC{$\vdash (\exists y a(y)) \otimes B, a(u)^\perp, B^\perp$}
\RightLabel{$\oplus_1$}
\UnaryInfC{$\vdash (\exists y a(y)) \otimes B, a(u)^\perp, B^\perp \oplus c(x)$}
\RightLabel{$\forall$}
\UnaryInfC{$\vdash (\exists y a(y)) \otimes B, \forall z a(z)^\perp, (B^\perp \oplus c(x))$}
\RightLabel{$\wp$}
\UnaryInfC{$\vdash (\exists y a(y)) \otimes B, (\forall z a(z)^\perp) \wp (B^\perp \oplus c(x))$}
\end{prooftree}
$$

Let us consider again the array of theorem 3.2, column by column, to see which inference types are well adapted to the forward movement. The result is that ,after analysis, the adequate inferences are $\wp, \&, c?, \forall, \perp$. On the contrary of what happens for the case of backward movement, they all have the same behavior (or almost, because of c? we are obliged to extend the notion of permutability of two inferences). Then we have only one theorem, illustrated by the above example. Apparently, the inferences of type w? and $\oplus$ could rank also among this category but we shall see at the end of the proof 4.5. why it is false.

Theorem 4.5 *(forward movement of* $\wp$*, &, c?,* $\forall$*,* $\perp$*)*
Let Π *be a proof in LL and I an inference of* Π *of type* $\wp$*, &, c?,* $\forall$*,* $\perp$*, there exists a proof* Π' *obtained by forward movement of I in* Π *such that the inference I' of* Π' *resulting of this movement is either the last inference of* Π' *or is followed by an inference with the principal formula of I as the active formula.*

Proof 4.5 *By structural induction on* Π*.*
We can restrict this proof to a proof Π *where the principal formula of I is not active afterwards (it is then in the conclusion of* Π*) without loosing generality.*
Let us denote I_l *the last inference of* Π*.*
1) $\underline{I \equiv I_l}$
The proof Π' *is equal to* Π*.*
2) $\underline{I \not\equiv I_l}$
2.1) $\underline{type(I_l) \neq \&}$*.*
We apply the induction hypothesis to the subproof Π_1 *of the premise of* I_l *that contains the principal formula of I.*
By forward movement of I in Π_1 *we obtain a proof* Π_1' *ending with an inference* I_1' *resulting of this movement.*
Let us replace Π_1 *by* Π_1' *in* Π*, by theorem 3.2,* I_1' *is permutable with* I_l *and we obtain the proof* Π' *through this permutation.*

2.2) $\underline{type(I_l) = \&}$.
Let F be the principal formula of I, Π has the following form:

$$\dfrac{\Pi_1\left\{\begin{array}{c}\vdots\\ \vdash F, G, \Delta\end{array}\right. \qquad \Pi_2\left\{\begin{array}{c}\vdots\\ \vdash F, H, \Delta\end{array}\right.}{\vdash F, G\&H, \Delta}$$

Let us assume, for example, that I is an inference of Π_1 and let J be the inference of Π_2 introducing F, we distinguish two cases, according to the relationship between the type of J and the type of I.
α) $\underline{F \not\equiv ?F'}$.
In this case, I and J have the same type and we apply the induction hypothesis to Π_1 with I and to Π_2 with J. By forward movement of I in Π_1 we obtain a proof Π'_1 ending with an inference I_1 resulting from this movement. In the same way, by forward movement of J in Π_2 we obtain a proof Π'_2 ending with an inference J_1 resulting from this movement. By replacing Π_1 and Π_2 by Π'_1 and Π'_2 respectively in Π and then by permutation of I_1 with I_l we obtain the proof Π'.
β) $\underline{F \equiv ?F'}$.
In this case, we apply the induction hypothesis on Π_1. By forward movement of I in Π_1, we obtain a proof Π'_1 ending with an inference I_1, resulting from this movement, that is necessarily of c? type. Let us replace Π_1 by Π'_1 in Π to obtain the following proof

$$\dfrac{\Pi'_1\left\{\dfrac{\left.\begin{array}{c}\vdots\\ \vdash ?F', ?F', G, \Delta\end{array}\right\}\Pi''_1}{\vdash ?F', G, \Delta}\,c?\right. \qquad \left.\begin{array}{c}\vdots\\ \vdash ?F', H, \Delta\end{array}\right\}\Pi_2}{\vdash ?F', G\&H, \Delta}\,\&$$

By introducing an inference of type w? at the end of Π_2, we can then permute I_1 with I_l to obtain the proof Π' (it implies an extension of the permutability notion but it is not a problem).

$$\dfrac{\dfrac{\Pi''_1\left\{\begin{array}{c}\vdots\\ \vdash ?F', ?F', G, \Delta\end{array}\right. \qquad \dfrac{\left.\begin{array}{c}\vdots\\ \vdash ?F', H, \Delta\end{array}\right\}\Pi_2}{\vdash ?F', ?F', H, \Delta}\,w?}{\vdash ?F', ?F', G\&H, \Delta}\,\&}{\vdash ?F', G\&H, \Delta}\,c?$$

Remark 4.1 *The following counter-examples illustrate well why the previous theorem can not be applied to the inferences of type $\oplus$ or w?.*

$$\dfrac{\dfrac{\dfrac{}{\vdash A, A^\perp}}{\vdash A, A^\perp \oplus B^\perp}\,\oplus_1 \qquad \dfrac{\dfrac{}{\vdash B, B^\perp}}{\vdash B, A^\perp \oplus B^\perp}\,\oplus_2}{\vdash A\&B, A^\perp \oplus B^\perp}\,\& \qquad \dfrac{\dfrac{\dfrac{}{\vdash A, A^\perp}}{\vdash ?A, A^\perp}\,? \qquad \dfrac{\dfrac{}{\vdash 1}}{\vdash ?A, 1}\,w?}{\vdash ?A, A^\perp\&1}\,\&$$

Concerning the inferences of type !, it seems difficult to move forward such inferences in a proof. It is true when we try to move it by successive permutations. But the ! inferences have the specific property, under some conditions, to be able to disappear in some points of a proof to appear again much further forward. It is pointed out in the following theorem

Theorem 4.6 *(forward jump of !)*
If a sequent of the form $\vdash !F, ?\Delta$ is provable then there exists a proof of this sequent which ends with the inference introducing !F.

Proof 4.6 *If a sequent of the form $\vdash !F, ?\Delta$ is provable and Π is a proof of it, we can modify it in the following way. At first, we replace, by starting from the final conclusion in all intermediate conclusions, the formula !F by F to the inferences which introduce it and we delete these inferences. Next, we add the following inference* $\dfrac{\vdash F, ?\Delta}{\vdash !F, ?\Delta}$ *at the end of the proof.*
It is clear that the resulting tree is definitely the proof we search.

A consequence of this theorem is a possibility of forward jump of ! in a proof.

Remark 4.2 *If we consider the inference types, we observe that we have different groups of inferences with respect to the permutability properties and the movement of an inference in a proof. We have two principal groups of inference types that are the ones to move down and the ones to move up. Let us remark that the partition between the both groups depends on the proof search direction (bottom-up or top-down) we choose, as we will see in section 6. Moreover, they correspond to the groups of connectives obtained by partition in [2] with respect to a notion of synchronization and determinism. But there are differences concerning the treatment of ? and !. The connective ?, considered as asynchronous by Andreoli [2], corresponds in our work to the three rules c?, w? and ? that behave differently. ! that is member of the group of synchronous connectives is moved forward in our work. Our own classification is based only on permutability properties in the framework and on their logical consequences. That is the reason why we do not tell about the constants* $\mathbf{0}$, $\mathbf{1}$ *and* $\top$ *that are produced by axioms.*

5 Redundancy reduction

The study of the forward and backward movement in the CLL proofs provides through the previous results and theorems tools to order the inferences inside a proof with a view to giving what can be called a normal form. Before defining this notion, let us treat a question that is strongly connected to this objective: how can we eliminate some reasons of redundancy in a proof ?

Definition 5.1 *A proof of CLL is* redundant *if two of its consecutive intermediate conclusions are identical.*

If, from a given proof, it is easy to construct an equivalent one which is not redundant, it is more difficult to construct directly, from a given sequent, a non-redundant proof (without needing to check it a posteriori).

5.1 Cut elimination

The cut rule is one of the reasons for redundancy. If the theorem 4.3. motivates the backward movement of cut inferences in a proof then it does not eliminate them. Let us consider this result on cut elimination as an extension of this theorem. Considering a proof of CLL, a cut can be moved backward up to the inferences introducing its active formulae. These ones have dual types and then we can move it again of one step. After this, we can apply again the theorem and so on... But how can we be sure that this process terminates ? Neither the height of the successive proofs nor the inference number decreases systematically during this process. Thus it is necessary to define an appropriate new decreasing function presented below, which is possible with the notion of complexity of cuts in a proof.

5.1.1 Complexity of cuts in a proof

Definition 5.2 *(formula complexity)*
Let Π *be a proof of CLL and* F *a formula occurrence in the conclusion of an inference* I *of* Π*, the* complexity of the formula F *the integer c(F) is defined inductively by:*
- *if* $F \in \Delta_p(I)$ *and if* $type(I) \in \{ax, w?, \perp\}$ *then* $c(F) = 1$.
- *if* $F \in \Delta_p(I)$ *and if* $\Delta_a(I) \neq \emptyset$ *then* $c(F) = 1 + max\{c(G), G \in \Delta_a(I)\}$.
- *if* $F \notin \Delta_p(I)$ *then* $c(F) = max\{c(F'),\ F'\ premise\ of\ I\ and\ F\ \equiv\ F'\}$.

Definition 5.3 *(cut complexity)*
Let Π *a proof of CLL and* I *a cut inference in* Π *the active formulae of which being* F *and* $F^\perp$*, the* complexity of the cut I *is the integer defined by* $c(I) = c(F) + c(F^\perp)$.

Until now, these definitions do not present difficulties, but the next one is more subtle. If we define the complexity of the cuts in a proof as the maximum of the complexity of its different cuts, it can be very difficult to prove that we can decrease this complexity. The reduction of the complexity of a particular cut could imply the increasing of another cut, being above. That motivates the following definition

Definition 5.4 *(cuts complexity in a proof)*
Let Π *be a CLL proof, the* complexity of the cuts *of* Π *is the integer* $c(\Pi)$ *that is equal to zero if* Π *has no cuts and to the maximum of the complexities of cuts* <u>*that are not preceded by another one.*</u>

5.1.2 Cut elimination

Theorem 5.1 *(complexity reduction)*
Let Π *be a proof in CLL of a sequent* $\vdash \Delta$ *with only one cut, there exists a proof* Π' *of* $\vdash \Delta$ *such that* $c(\Pi') < c(\Pi)$.

Proof 5.1 *The complete proof is given in appendix B.*

The cut elimination theorem is a direct consequence of the theorem 5.1, but its proof uses a particular strategy of cut reduction and thus our theorem is a theorem of weak normalization knowing that strong normalization is verified for linear logic [12].

Theorem 5.2 *(cut elimination)*
If $\vdash \Delta$ *is a provable sequent of CLL then there exists a proof of* $\vdash \Delta$ *without cuts.*

Proof 5.2 *Let* Π *be a proof of CLL, for* $n \in [0, c(\Pi)]$ *we have the property*
P(n): there exists a proof Π_n with the same conclusion as Π such that $c(\Pi_n) = c(\Pi) - n$.
This property is an immediate consequence of the theorem 5.2. It is sufficient then to apply $P(c(\Pi))$.

Theorem 5.3 *(of subformula)*
Let Π *be a CLL proof without cuts and* $\vdash \Delta$ *its conclusion,*
for any intermediate conclusion $\vdash \Delta'$ *of* Π*, a formula of* $\vdash \Delta'$ *is a subformula of* $\vdash \Delta$*.*

Proof 5.3 *By induction on h, height of* $\vdash \Delta'$ *in* Π*.*
• *h = 0. In this case* $\vdash \Delta'$ *is* $\vdash \Delta$ *and the property is true.*
• *Let us assume the property true for any order h and show that it is true for h+1.*
Let $\vdash \Delta'$ *be an intermediate conclusion of* Π *being at the height h+1, it is a premise of an inference I the conclusion of which is at height h.*
Let F be a formula of $\vdash \Delta'$*, I is not a cut and then there exists a formula* F' *of a sequent* $\vdash \Delta''$ *such that F is a subformula of* F' *(extending the subformula notion for* $\exists$ *and* $\forall$*).*
$\vdash \Delta''$ *being at height h in* Π *, the induction hypothesis is applied to it and* F' *is a subformula of a formula* F'' *of* $\vdash \Delta$*.*
Then, by transitivity, we deduce that F is a subformula of F''*.*

Let us recall that this theorem of sub-formula is naturally the basis of proof search methods in bottom-up approaches but also in top-down approaches.

5.2 Weakening and contraction reduction

The cuts often generate redundancy in a proof but it is not the only reason. For example, the following proof

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash A, A^{\perp}}}{\vdash ?A, A^{\perp}}}{\vdash ?A, ?A, A^{\perp}}}{\vdash ?A, A^{\perp}}$$

is redundant but does not contain cuts. It seems that it is connected to weakening and contraction rules that interact.

Definition 5.5 *A proof* Π *of CLL is* under weakening and contraction reduction *if for any intermediate conclusion under the form* $\vdash ?F, ?F, \Delta$*,*
(i) if it is the conclusion of a w? rule introducing a formula ?F then there exists an inference J ($\neq$ *c?) introducing the other formula ?F.*
(ii) if it is the premise of a c? rule with ?F being one of its active formulae then, for each formula ?F, there exists an inference J ($\neq$ *w?) that introduces it.*

Remark 5.1 *In the case where the weakening is done immediately after the axioms the condition (i) is always true and then (ii) is the only condition to satisfy.*

Theorem 5.4 *For any CLL proof* Π *with* $\vdash \Delta$ *as conclusion there exists a proof* Π' *of* $\vdash \Delta$ *under weakening and contraction reduction that is obtained by elimination of some weakening or contraction inferences and also addition of some intermediate conclusions.*

Proof 5.4 *By induction on h, height of the proof* Π*.*
• *h = 0. There is no proof* Π *with* $h = 0$ *and thus the property is true.*

• *Let us assume that we have the result for any proof tree with height h et prove it for h+1. Let* Π *be such a proof tree, if it is under weakening and contraction reduction we have the expected result, if not it contains an intermediate conclusion* $\vdash \Delta_1$ *of the form* $\vdash ?F, ?F, \Delta$ *that does not verify the condition (i) and (ii) of definition 5.5.*
α) Δ does not verify the condition (i).
We suppress in Π *the weakening the conclusion of which is* $\vdash \Delta_1$ *and all contractions introducing one of the formulae ?F and we add ?F to intermediate conclusions situated between them. Thus, we obtain a proof* Π'' *of height h-1 with the same conclusion as* Π *and we can apply the induction hypothesis to it.*
β) Δ does not verify the condition (ii). We use the same proof schema as in α.

Theorem 5.5 *A proof of CLL without cuts and contractions is not redundant.*

Proof 5.5 *This proof is based on the fact that for each inference the conclusion contains a number of logical connectives greater than the one of each premise.*

Unfortunately, the theorem is false if we allow the contractions. Here we give an example illustrating this point.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash A, A^{\perp}} \quad \dfrac{}{\vdash A, A^{\perp}}}{\vdash A \otimes A^{\perp}, A, A^{\perp}}\;\otimes}{\vdash ?(A \otimes A^{\perp}), A, A^{\perp}}\;? \quad \dfrac{}{\vdash A, A^{\perp}}}{\vdash ?(A \otimes A^{\perp}), A \otimes A^{\perp}, A, A^{\perp}}\;\otimes}{\vdash ?(A \otimes A^{\perp}), ?(A \otimes A^{\perp}), A, A^{\perp}}\;?}{\vdash ?(A \otimes A^{\perp}), A, A^{\perp}}\;c?$$

This proof is redundant and without cuts. It is even under weakening and contraction reduction. That shows that we have not exhausted the subject of redundancies elimination in the proofs.

6 Proof normalization

From the previous results about the permutability of inferences, the movements of an inference in a proof and the treatment to reduce some redundancies, we aim to analyze the possibilities of reducing the non-determinism of the proof search process. In this section, we study the proof normalization depending on the strategy we use for proof construction. We will see in a next section that it might depend also from the fragment of CLL we consider. Thus, we are able to define the notion of normal proof that is a special form of proof with certain constraints on the order in which the inference rules must be applied. It is a way to consider the mechanization of proof construction in full linear logic, these normal proofs constituting a complete subset of proofs in CLL. But it would be necessary, in further work, to consider in addition semantical properties to improve new strategies for theorem proving in CLL.

6.1 Normalization and construction strategy

Undependently of the construction strategy, the possibility to construct proofs without cuts is essential. For a top-down proof search, it allows an goal-oriented procedure because of the

sub-formula property and for a bottom-up proof search we have a decomposition process of the formulas of the sequent to prove.
The theorems about the inference movements lead to a relative ordering of the inferences in a proof. Considering a proof as an object, the normalization corresponds to a double movement: maximum up or down movements of certain inferences in a proof. Considering a proof more as a process (being interested by the construction), the normalization corresponds to the application as soon as possible of certain inferences and as late as possible of other inferences.
The theorems of section 4 help us to determinate a first criterion to choose the inferences to move backward (up) or forward (down): *the facility to do it.*
For example, theorem 4.5. allows to immediately decompose a formula having & is principal connector when it appears in a goal during a bottom-up proof search. In the contrary, for a top down search, the application of the & rule is done as late as possible. But this criterion about facility is the not the only one. Another one consists of the proof strategy we have chosen (bottom-up or top-down). Thus, for top-down search, we have to move up the inferences that are easy to control from the premises and to move down at maximum the others. For a bottom-up, we have to move down the inferences that are easy to control from the conclusion and to move up at maximum the others.
For instance, if we consider the weakening rule $w?$, we observe that it is not controllable from the premise and from the conclusion. Thus, we have to apply this rule as late as possible for top-down and bottom-up proof directions.
The analysis for the contraction rule $c?$ is different because it is not controllable from the conclusion but more easier from the premise. Thus, we have to apply the rule as soon as possible for a top-down strategy and as late as possible for a bottom-up strategy.
Let us consider the following example

$$\cfrac{\cfrac{\cfrac{\cfrac{\vdots}{\vdash F, ?G, \Delta_1} \qquad \cfrac{\vdots}{\vdash H, ?G, \Delta_2}}{\vdash F \otimes H, ?G, ?G, \Delta_1, \Delta_2}\ \otimes}{\vdash F \otimes H, ?G, \Delta_1, \Delta_2}\ c?}{\vdash (F \otimes H) \oplus K, ?G, \Delta_1, \Delta_2}\ \oplus$$

We have, for a bottom-up approach, moved up $c?$ as far as possible but in this case, we do not apply the *focusing* principle [2, 10] that means here that if we decompose the formula $(F \otimes H) \oplus K$ we have to go on with it until $\otimes$ or $\oplus$ is not the principal connective (it is not the case in this example). Thus, if we want to keep the movement of $c?$ compatible with thus principle, we can modify the $\otimes$-rule to the following

$$\cfrac{\vdash F, \Gamma_1, ?\Delta \qquad \vdash G, \Gamma_2, ?\Delta}{\vdash F \otimes G, \Gamma_1, \Gamma_2, ?\Delta}\ \otimes'$$

This rule is easily derivable from the initial $\otimes$ rule and is also complete and it is easy to prove that the sequent modified calculus with $\otimes'$ preserves completeness [27]. Thus, from now, we consider this new version of the sequent calculus.
When we have fixed the direction of an inference movement in a proof, we have to known until where this movement is possible. The general answer is simple: in the proof (as object) the inferences will be moved up until the inferences where the active formulas are introduced and moved down until the inferences where their principal formulas become active.
That are consequences of the results in section 4. Moreover, we have to consider the coherence of these movements. To be more concrete, we want now consider the bottom-up construction

of proof in CLL, aiming the analysis the non-determinism forms and their reasons and the reduction possibilities due to the normalization. A similar and dual study might be done for CLL and the top-down proof strategy.

6.2 Normalization and non-determinism reduction

Let us consider a sequent $\vdash \Delta$ to prove in CLL, a bottom-up proof search consists in building the proof tree from the conclusion to the axioms. The process is a construction of a sequence of goal expansions, an expansion of a goal $\vdash \Delta'$ consisting in replacing it by the premises of an inference that has $\vdash \Delta'$ as conclusion. During each step, there are three fundamental selection choice points where we have non determinism: the choice of a goal (to satisfy), the choice of a principal formula in a goal and finally, from a given goal and a principal formula, the choice of an inference with this goal as conclusion and this formula as the principal one.
a) The choice of a goal.
The non-determinism concerning this choice is a "don't care" non-determinism in the sense that whatever the goal we choose, the result does not change as its form. It is true for normal proofs or not. But the choice has consequences on time and space resources used to determine the goal. The study about permutability of inferences in a proof and the resulting theorem 3.2 can help us to elaborate some strategies. A sequent is often an element of a particular fragment of CLL and the permutation of inferences (and then the normalization) can be done more efficiently depending on the sort of fragments. Then it is pertinent to consider the goals being in fragments where we expect good normalization properties. Other considerations, from a semantical point of view, can help to refine the goal choice. For example, a goal being in the multiplicative fragment of CLL is interesting because we can apply the duality property [11] to it.
b) The choice of a principal formula.
This choice concerns partially a "don't know" non-determinism and a "don't care" non-determinism. The proof normalization will allow to reduce it directly.

- if the goal has the form $\vdash !F, ?\Delta'$ then theorem 4.6 allows us to surely choose $!F$ as principal formula of an inference of ! type and to replace the current goal by $\vdash F, ?\Delta'$.
- if the goal includes a formula having $\wp, \&, \forall, \perp$ as principal connective then we can surely choose it as principal formula of the next inference that will reduce the goal (theorem 4.5).
- if the goal does not have the previous forms and if it contains formulas that are not literals, the choice of the principal formula corresponds to a "don't know" non-determinism. But when we fix the principal formula F, we determinate the principal formula of the inferences that immediately follow. If F has $\oplus, \otimes, \exists$ as principal operator, we can surely choose its components, when they are not positive literals or not of the form $?G$, as principal formulas of the inferences that immediately follow (theorem 4.4).
  If F has the form $?G$, it can be produced by three different rules $c?$, $w?$, and ?. We surely choose an inference of $w?$ type if the goal can be proved by an axiom followed by a sequence of weakenings (theorem 4.1). Else, if F is produced by the ? rule, by theorem 4.2. we can choose the active formula as principal formula of the next inference and if F is produced by the $c?$ rule, one of the two active formulas is the principal formula of the inference that immediately follows.
- if the goal has only literals, it is either not provable or it is the conclusion of an axiom.

c) <u>The choice of an inference</u>
Having a principal formula, its external connective determines the inference rule to realize the expansion of the proof tree, except in the case of ? and also $\oplus$ for which we have two possibilities. The choice between $\oplus_1$ and $\oplus_2$ corresponds to a "don't know" non-determinism that is difficult to reduce.
The rule being fixed, it does not mean that the corresponding inference is completely fixed. It is true in general, except for $\exists$ and $\otimes$. For $\exists$, we have to choose the term associated to the inference. For $\otimes$, we have to split the context into two parts that is an important source of "don't know" non-determinism difficult to reduce. In the both cases we can postpone the choice at the level of axioms by using lazy methods.
To summarize, the search of a normal proof imposes constraints to the general algorithm for the choice of the principal formula, that reduce significantly the "don't know" non-determinism of it. Moreover, it could help us to elaborate specific tactics and strategies for the choice of the goal to prove. But it does provide mechanisms for calculating expansions when they are not determined by the principal formula. Thus, normal proofs provide logical foundations for the proof search for a given sequent.

6.3 Normal proofs

To the normalization of proofs considered as processes, corresponds a normalization of the proofs considered as objects. For the full linear logic and for the bottom-up proof search direction, we can now define the notion of *normal proof*.
Let us begin to introduce this definition, by considering $T_\downarrow$ and $T_\uparrow$ the sets of inference types defined by $T_\downarrow = \{\wp, \&, \forall, \bot\}$ and $T_\uparrow = \{\otimes, \oplus_1, \oplus_2, c?, w?, ?, \exists\}$.

Definition 6.1 *A proof* Π *of CLL is said* normal *if*
(i) Π *is without cuts,*
(ii) Π *is under weakening and contraction reduction,*
(iii) any intermediate conclusion $\vdash \Delta$ *verifies:*
if $\vdash \Delta$ *has the form* $\vdash !F, ?\Delta'$
then *it is the immediate conclusion of an inference of* ! *type*
else if $\vdash \Delta$ *contains a formula introduced by an inference of type* $\in T_\downarrow$
    then *it is the immediate conclusion of an inference of type* $\in T_\downarrow$
    else if $\vdash \Delta$ *contains a formula introduced by an inference of type* $\in T_\uparrow \setminus \{?, c?, w?\}$
        then *for each premise which is not* $\vdash !F, ?\Delta'$*, the active formula*
        , if it is not a positive literal, is the principal formula of the preceding inference.
        else if $\vdash \Delta$ *has a formule* $?F$ *as principal formula*
            then *we have three possible cases*
                (1) $\vdash \Delta$ *is the conclusion of a* $w?$ *rule and the preceding inferences are weakenings or axioms.*
                (2) $\vdash \Delta$ *is the conclusion of a* $c?$ *rule and the preceding inference is an inference of type* ? *introducing one of the active formulas of the inference ending with* $\vdash \Delta$
                (3) $\vdash \Delta$ *is the conclusion of an inference of* ? *type and the active formula, if it is not a positive literal, is the principal formula of the preceding conclusion.*

An intermediate conclusion which verifies this criterion is said normal.

Example 6.1 *The following proof*

$$
\begin{prooftree}
\AxiomC{}
\UnaryInfC{$\vdash A, A^\perp$}
\RightLabel{$\oplus_1$}
\UnaryInfC{$\vdash A \oplus B, A^\perp$}
\AxiomC{}
\UnaryInfC{$\vdash B, B^\perp$}
\RightLabel{$\oplus_2$}
\UnaryInfC{$\vdash A \oplus B, B^\perp$}
\RightLabel{$\&$}
\BinaryInfC{$\vdash A \oplus B, A^\perp \& B^\perp$}
\AxiomC{}
\UnaryInfC{$\vdash C, C^\perp$}
\RightLabel{$\otimes$}
\BinaryInfC{$\vdash (A \oplus B) \otimes C, A^\perp \& B^\perp, C^\perp$}
\RightLabel{$\wp$}
\UnaryInfC{$\vdash (A \oplus B) \otimes C, (A^\perp \& B^\perp) \wp C^\perp$}
\AxiomC{}
\UnaryInfC{$\vdash D, D^\perp$}
\RightLabel{$\otimes$}
\BinaryInfC{$\vdash ((A \oplus B) \otimes C) \otimes D, (A^\perp \& B^\perp) \wp C^\perp, D^\perp$}
\end{prooftree}
$$

is not in normal form because it contains intermediate conclusions which are not normal, for example $\vdash (A \oplus B) \otimes C, A^\perp \& B^\perp, C^\perp$*. It contains a formula introduced by an inference of type* $\in T_\downarrow$ *(i.e.,* $A^\perp \& B^\perp$*) and it is the immediate conclusion of an inference of type* $\in T_\uparrow$ *(i.e.,* $\otimes$*).*
For the same sequent to prove, the following proof is in normal form

$$
\begin{prooftree}
\AxiomC{}
\UnaryInfC{$\vdash A, A^\perp$}
\RightLabel{$\oplus_1$}
\UnaryInfC{$\vdash A \oplus B, A^\perp$}
\AxiomC{}
\UnaryInfC{$\vdash C, C^\perp$}
\RightLabel{$\otimes$}
\BinaryInfC{$\vdash (A \oplus B) \otimes C, A^\perp, C^\perp$}
\AxiomC{}
\UnaryInfC{$\vdash D, D^\perp$}
\RightLabel{$\otimes$}
\BinaryInfC{$\vdash ((A \oplus B) \otimes C) \otimes D, A^\perp, C^\perp, D^\perp$}
\AxiomC{}
\UnaryInfC{$\vdash B, B^\perp$}
\RightLabel{$\oplus_2$}
\UnaryInfC{$\vdash A \oplus B, B^\perp$}
\AxiomC{}
\UnaryInfC{$\vdash C, C^\perp$}
\RightLabel{$\otimes$}
\BinaryInfC{$\vdash (A \oplus B) \otimes C, B^\perp, C^\perp$}
\AxiomC{}
\UnaryInfC{$\vdash D, D^\perp$}
\RightLabel{$\otimes$}
\BinaryInfC{$\vdash ((A \oplus B) \otimes C) \otimes D, B^\perp, C^\perp, D^\perp$}
\RightLabel{$\&$}
\BinaryInfC{$\vdash ((A \oplus B) \otimes C) \otimes D, A^\perp \& B^\perp, C^\perp, D^\perp$}
\RightLabel{$\wp$}
\UnaryInfC{$\vdash ((A \oplus B) \otimes C) \otimes D, (A^\perp \& B^\perp) \wp C^\perp, D^\perp$}
\end{prooftree}
$$

We present now the proof normalization theorem, the proof of which gives us a procedure for the transformation and normalization of given proofs.

Theorem 6.1 *(normalization theorem)*
For any proof Π *of CLL there exists a normal proof* Π_n *of LL with the same conclusion*

Proof 6.1 *Let us call a* quasi-normal proof *a proof that verifies all normality criteria of the definition 6.1 except possibly (ii).*
According to theorem 5.4, it is sufficient to prove that, for any CLL proof there exists a quasi-normal proof with the same conclusion.
Let Π *be a CLL proof, according to theorem 5.2, we can also suppose* Π *without cuts. In order to normalize it, we can choose between different strategies. We can, for example, start with the forward movement of all concerned inferences and then move backward all the others. Of course, we must proceed methodically and carefully because every new movement does not have to destroy what has been realized before. Then, we proceed by structural induction on* Π.
1) <u>Π *is an axiom.*</u>
In this case, Π *is obviously normal and is the proof we search.*
2) <u>Π *is not an axiom.*</u>
Let I be the last inference of Π*. By the induction hypothesis, we can replace the subproofs of the I premises by quasi-normal proofs in* Π*. We obtain in this way a proof* Π' *which is not necessarily quasi-normal. The conclusion* $\vdash \Delta$ *of I is not always a normal intermediate conclusion of* Π'*.*
It depends on the type of I.
2.1) <u>*type*$(I) = !$</u>
It is immediate that $\vdash \Delta$ *is normal in* Π'*.*
2.2) <u>*type*$(I) \in T_\downarrow$</u>
Then $\vdash \Delta$ *is not of the form* $\vdash !F, ?\Delta'$ *and therefore it is normal in* Π'*.*

2.3) $\underline{type(I) \in T_\uparrow \setminus \{?, w?, c?\}}$

According to the theorem 4.4, there exists a proof Π''*, obtained by backward movement of I in* Π'*, such that every inference I' of* Π'' *resulting from this movement is immediately preceded by an inference which introduces the active formula of I' (if its top connective is not ?).*
This operation preserves the normality of the modified intermediate conclusions, except possibly the conclusions of the inferences I'. They can contain formulas which have been introduced by inferences of type $\in T_\downarrow$*. But we can modify this feature using theorem 4.5 which allow, for every inference I', to move forward the problematic inferences just after I'.*
The proof we obtain is then quasi-normal.

2.4) $\underline{type(I) \in \{?, w?, c?\}}$

We use the same process as in 2.3) but with variants due to particularities of these types. In its backward movement, I cannot be stopped by an inference of ! type and according to the theorem 4.1, I can be moved just after axioms if it is of w? type.

Remark 6.1 *The concept of normal proof, because of its static character, does not emphasize the essential difference between the inferences of type* $\wp, \&, \forall, \top$ *and the ones of type* $\oplus, \otimes, \exists, ?$*. The first ones are completely concerned by a "don't care" non-determinism and the second ones are partially concerned by a "don't know" non-determinism.*

Remark 6.2 *Moreover, the existence of a part of "don't care" non-determinism leads to different normal proofs for a given sequent, each being equivalent modulo inference permutations. A possibility for handling this specific case of inference permutation could be to investigate a notion of proof net and proof net normalization. Proof net is a concept in proof theory firstly introduced by Girard for the multiplicative fragment of linear logic [12] from the fact that the sequential presentation of proofs with trees is inadequate and does not emphasize their meaning. In [11], we have investigated automatic proof net construction but the point is to be able to extend this notion to more important fragments of CLL [5] through a new appropriate representation and definition of proof nets. Having it, it would be interesting to apply the previous results on normalization directly on this concept with a view to reducing not useful redundancies.*

As in [16] where the *uniform proof* notion is essential for proof construction, we have with the *normal proof* concept the possibility to found systems dependent from proof search in CLL on this concept defining sub-classes of proofs that are complete and tractable.
It is important to note that this notion depends on the fragment we consider and also on the proof construction direction (bottom-up or top-down). In a general way, if we want to build normal proofs in CLL it is necessary to fix some constraints about choices, as mentioned above, with effect to mainly reduce non-determinism sources.

7 Application to linear logic programming

Until now, we have considered full linear logic and normalization aims to improve the efficiency of proof construction in a theorem prover for linear logic. Such a prover could be considered as a starting point for a based-on-linear-logic interpretor of logic programming. But it seems necessary to restrict the study to some fragments of CLL for efficiency purposes. Thus a crucial question arises: how can we determine an appropriate fragment ? .
A first point to answer this question is to know the type of problems we want to specify with the help of linear logic. Commonly, it is well adapted to the specification of dynamic problems where

we need a strict and explicit resource management as planning [23], natural languages analysis, Petri nets and more generally reactive systems. Thus, for a given problem, we can reduce the syntax to the one necessary for coding it as for Petri nets where only the $\{\otimes, ! \multimap\}$ fragment is involved. At this point, nothing tells us if the fragment determined by the problem involves efficient proof search procedures. The previous results will help us to analyze the adequacy between the fragment of logic and the possible proof search methods.
To illustrate the application to the design of linear logic programming, including efficient proof search procedures, we consider the example of the representation of the standard Prolog in linear logic and the analysis of possible proof procedures. Let us note that this example is only significant for the illustration of the methodological point of view but applications of this approach to other logic programming proposals surely will emphasize its foundational character.

Let us consider the logical fragment of standard Prolog, i.e, Horn clauses.
Basically, we have a program P and a goal G composed by clauses, each clause C_i of P being of the form $A_i \Leftarrow B_1, B_2, .., B_n$.

1) A first step is the translation of the program and goal clauses.
each clause C_i is translated in CLL by the formula $C_i^L \equiv !(\forall x_1, .., x_m(B_1 \& B_2 \&, ..\& B_n \multimap A))$ where the $x_i (1 \leq i \leq m)$ are the free variables of C_i.
A goal G is translated by the formula $G^L \equiv \exists y_1, .., y_p(G_1 \& G_2 \& ... \& G_q)$ where the $y_j(1 \leq j \leq p)$ are free variables of G and G_l are subgoals of G.
Thus the general query $< P, G >$ will be represented by the following linear sequent

$$C_1^L, .., C_k^L \vdash G^L.$$

To summarize we can say that the Prolog queries correspond to sequents in Linear Logic of the form $!C_1, .., !C_k \vdash \exists x_1, .., x_p G$ where C_1,..,C_k are particular formulas of CLL representing the clauses of the program and G is a formula representing the goal of the query, both being defined by the following grammar

$$C := X | G \multimap X | \forall x C.$$
$$G := X | G \& G | \exists x G.$$

where X represents a positive literal, C a definite clause and G a goal.
Let us remark that here, we have translated a given logical framework to obtain the previous grammar about clauses and goals. That is because we have chosen standard Prolog as starting point, but a normal approach is to give directly such a grammar and to begin the study with it.

2) A second step consists in defining the CLL fragment involved by this sequent form.
Thus, according to the subformula property we can deduce, from the previous syntax, the set of inference rules of CLL that will be concerned to prove the linear sequents deduced by the first step. For our example, this set of rules is $R_0 = \{\multimap_L, \&_R, !_L, w!_L, c!_L, \forall_L, \exists_R\}$ and we have now determinated the logical fragment (denoted LF_0).

3) A third step consists in studying the logical foundations of this fragment, i.e., permutability properties in order to define proof search procedures.
Let us remark that our results work on sequents without left-hand side part but we can transpose them without difficulty for application to classical sequents. The study of inference permutability, according to theorem 3.2, leads to the following results for LF_0, summarized in the array below.

$\mathbf{t_2 \backslash t_1}$	$-\!\circ_L$	$\&_R$	$!_L$	$w!_L$	$c!_L$	$\forall_L$	$\exists_R$
$-\!\circ_L$	p	p	p	p	p	p	p
$\&_R$	np	$\times$	np	np	np	p	$\times$
$!_L$	p	p	p	p	p	p	p
$w!_L$	p	p	p	p	p	p	p
$c!_L$	np	p	p	p	p	p	p
$\forall_L$	p	p	p	p	p	p	p
$\exists_R$	p	$\times$	p	p	p	p	$\times$

4) A fourth step consists in studying the possible inference movements in LF_0. To do that, we have first to fix a direction for proof strategy, i.e, bottom-up or top down, and according to our example, we choose a bottom-up proof search strategy and try to order inferences as in the general case presented section 4.
For LF_0, the inferences of type $\&_R$ and $\exists_R$ can be moved forward (or down) as far as possible in a proof and the inferences of type $-\!\circ_L$, $!_L$, $w!_L$ and $c!_L$ can be moved forward (or up) as far as possible. Let us remark that $\forall_L$ can be moved up or down and we decide here to move it backward.

5) A fifth step consists in defining, through inferences ordering, the form of bottom-up proof we can construct.
From the previous step, a bottom-up proof will begin with the application of the $\exists_R$ and $\&_R$ rules for decomposing the goal into subgoals.
Then, we will apply the $!_L$ or $c!_L$ rules that correspond to the choice of a clause in a program. If we apply the $!_L$ rule, we can always have a $c!_L$ rule application preceding it and if it is useless we can cancel it by a weakening. It means that the set of resources of the program will be unchanged until the next step. The backward movement of $c!_L$ inferences means that the application of such a rule will be immediately followed by $!_L$ rule application (the application of the $w!_L$ rule could introduce a redundancy that we can suppress).
With a selected clause being usable through the application of $!_L$ rule we can apply it the $\forall$-L rule because the $!_L$ rules have been moved up at maximum.
Then we consider the $-\!\circ_L$ because the $\forall_L$ rules have been moved up at maximum and we have a goal of the form $!\Gamma, G -\!\circ X \vdash Y$.
In the classical strategy of Prolog we have X and Y unifiable but here we have X and Y identical. Let us explain why. The inference to realize has the form

$$\frac{!\Gamma_1 \vdash G \quad !\Gamma_2, X \vdash Y}{!\Gamma_1, !\Gamma_2, G -\!\circ X \vdash Y}$$

We know that it is moved up at maximum and thus the active formula X is also the principal formula of the inference just above, even it is an atom. X is an atom and this inference is an axiom and consequently $X = Y$ and $\Gamma_2 = \emptyset$ and the inference is

$$\frac{!\Gamma \vdash G \quad X \vdash X}{!\Gamma, G -\!\circ X \vdash X}$$

Here, we have partially the Prolog mechanism without unification because the corresponding rules $\forall$-L and $\exists$-R have been applied.
To obtain unification, it is sufficient to have a lazy application of these rules where the instantiation of the variables is stopped until the axioms application.

We have proposed a simple example, for which we have knowledge about expressiveness and proof search, but we can apply this analysis method to different fragments of linear logic as in [15, 16, 17], considered as basis for linear logic programming.

8 Related and further work

This work on proof normalization in linear logic presents similarities and differences with other works on various fragments of LL, mainly those focusing on extensions of logic programming.
The study of permutability properties is significant for proof search and theorem proving for non classical logics in general [29]. Shankar has presented in [26] a proof search method for intuitionistic calculus, based on the permutability possibilities, that could be generalized for our purpose. We will investigate this point and the connections with our work.
Our aim, here, consists in having a special proof form, called normal form, in the class of equivalent cut-free proofs. In a similar way, Andreoli emphasizes in [2] also a subclass of proofs, called "focusing proofs", which is complete. Let us recall that, in a normal proof, the weakenings ($w?$) are moved just after the axioms, the inferences of type $\wp, \&, \perp, \forall$ are moved forward as far as possible (until the principal formula becomes active or to the end of the proof), the inferences of type $\otimes, \oplus, ?, \exists, c?$ are moved backward as far as possible (until its active formulae are introduced). So we obtain a partition of inference types into two groups with the respect of the notion of movement of an inference in a proof. [2] proposes a similar partition of the connectives, consequently of the inference rules, that is based on notions of synchronization and determinism.
It presents two main differences concerning the treatment of connectives ? and !. In [2] Andreoli considers the connective ? as synchronous, which seems justified by its triadic system. But, the syntax of this system masks the point that the three types of inferences introducing ?, $w?$ and $c?$ have no common behaviour which our study confirms.
The difference about the treatment of ! is more important. He considers the connector ! as those he called "synchronous" connectives (for example, $\otimes$) and does not reduce the non-determinism in this case. Then a principal formula $!A$ is chosen with a non-determinism. But it is possible to suppress it because, as shown in the previous sections, in a sequent $\vdash !A, ?\Delta$, the formula !A can always be selected as principal. For example, if we have to prove $\vdash !A, ?(A^\perp \oplus B), ?B^\perp$ in [2] it is possible to choose $?(A^\perp \oplus B)$ as principal formula, then $?B^\perp$ before the right one $!A$ whereas in our system we choose immediately !A as principal formula.
Tammet in [27] concentrates on problems of automated theorem proving in full linear logic and investigate general search strategies mainly for top-down direction with original proposals for resolution method in CLL. It appears that we might consider our approach to refine and define proof strategies for top-down direction.
Keeping full linear logic for the proof normalization process, leads to some limits due to the impossibility to permute some inferences. A way to solve this problem is to consider an adequate CLL fragment to go further than in CLL in the inference movement in a proof and then in the normalization. For example, in a fragment without &, the inferences of type $\oplus$ can be moved forward as much as possible as those of type $\wp$. Of course, for the choice of the fragment, there is another important criterion to take into account: the ability to express problem specifications in such a fragment. Both requirements of expressiveness and efficiency can be contradictory and we have to find the best compromise between these aspects.
In this way, we can mention the work of Hodas and Miller [16] and of Harland and Pym [14, 15]. The common objective is to extend the expressiveness capability in logic programming languages

using linear logic and to efficiently construct proofs in the logical framework. They consider a two-sided linear sequent calculus without negation rule.
But are, in such a framework, our permutability properties still available ? The answer is yes if we made a good translation of it. Then any property of an inference type (for sequents without left-hand side part) is transposed automatically to the corresponding inference type at right-hand side and also to the dual inference type at left-hand side (see section 7). It results from the fact that any rule of the sequent calculus without left-hand side part leads to two rules when we consider a sequent calculus with it. Moreover, the connective $\multimap$, more adequate for logic programming, replaces the connective $\wp$ but keeps the same properties of inference permutability (in fact $A \multimap B \equiv A^{\perp} \wp B$).
Briefly, we can say that [16] considers two sorts of formulae (as goals and resources) and sequents (as queries) of a specific CLL fragment. The point is that such sequents can be proved using the notion of *uniform proofs*: in a bottom-up construction, the right rules are always applied before the left rules. It is the case because, in this fragment of LL, the inferences on right-hand side parts of sequents can be moved down as much as possible in a proof. This point is strongly connected to the previous results. A complete study could be done on the basis of the method proposed in section 7 to understand and justify the limits and the power of the fragment considered. Moreover, a similar study can be done with the approach of Harland and Pym [15]. They have proposed also a fragment of LL chosen so that uniform proofs remain complete but it presents more difficulty to treat right-hand sides including !, with difference of expressiveness in goals and contexts. These works refer to the resolution on CLL fragments which appears as a specific rule defined mainly from an analysis of the permutation properties as but with a bottom-up proof direction.
Finally, we cannot forget to mention the relationship with the fundamental results about complexity and decidability in LL. Kanovich's works [17] aim to develop a computational interpretation of the logic and to obtain efficient decision algorithms based on a bottom-up approach. To do it, he considers the Horn fragment of LL from a computational and a logical point of view and then generalizes the approach by introduction of the additives and !. Knowing that the propositional linear logic is not decidable [21], the main conclusions about this complexity analysis is that the multiplicative fragment is NP-complete and the additive multiplicative one is Pspace-complete. The connections between our based-on-permutability logical study and the various related work presented here are to be deeply analyzed with a view to mechanizing proof construction in fragments of linear logic.

9 Conclusion

We have considered the problem of proof normalization in full linear logic. The solution we propose results from a systematic study of inference permutabilities in this logic framework. An issue of them is the effective construction of a theorem prover for linear logic in which it is possible to reduce some sources of undeterminism during the proof construction. It would be necessary to consider other tactics or strategies for proof development based in addition on semantical results. Another issue consists in using the analysis on permutability of inferences in designing some logic programming languages in fragments of CLL with a compromise between the expressiveness of the language and the efficiency of the proof construction. This point is strongly connected to recent works on some extensions of logic programming [15, 16] and allows to understand or justify some choices in the language conception. Moreover, from this analysis of proof mechanization in linear logic and the better comprehension of CLL through this proof-theoretic foundation, we could consider the various applications of proof development in linear logic. In addition to logic programming, let us mention the connections with some dynamical problems as planning [22] and with the proofs as programs approach through adequate typed λ-calculi. In this latter case, a good knowledge about various aspects of proof construction and transformation would be necessary to use the algorithmic contents of proofs to prospect for example typed concurrent programming and program synthesis through program extraction from proofs in this logical framework.

References

[1] S. Abramsky. Computational interpretations of linear logic. *Theoretical Computer Science*, 111(1-2):3–58, 1993.

[2] J.M. Andreoli. Logic programming with focusing proofs in linear logic. *Journal of Logic and Computation*, 2(3):297–347, 1992.

[3] J.M. Andreoli and R. Pareschi. Logic programming with sequent systems: A linear logic approach. In *Int. Workshop on Extensions of Logic Programming, LNCS 475*, pages 1–30, Tübingen, Germany, December 1989.

[4] J.M. Andreoli and R. Pareschi. Linear objects: logical processes with built-in inheritance. In *7th Conference on Logic Programming, MIT Press*, pages 495–510, Jerusalem, June 1990.

[5] G. Bellin. Proof nets for multiplicative and additive linear logic. Technical Report ECS-LFCS 91-161, Department of Computer Science, Edinburgh University, May 1991.

[6] S. Cerrito. A linear semantics for allowed logic programs. In *5th IEEE Symposium on Logic in Computer Science*, pages 219–227, Philadelphia, June 1990.

[7] J. Gallier. *Logic for Computer Science: Foundations of Automatic Theorem Proving.* John Wiley & Sons, 1986.

[8] D. Galmiche. Constructive system for automatic program synthesis. *Theoretical Computer Science*, 71(2):227–239, 1990.

[9] D. Galmiche. Program development in constructive type theory. *Theoretical Computer Science*, 94(2):237–259, 1992.

[10] D. Galmiche and G. Perrier. Automated deduction in additive and multiplicative linear logic. In *Logic at Tver '92, Logical Foundations of Computer Science Symposium, LNCS 620*, pages 151–162, Tver, Russia, July 1992.

[11] D. Galmiche and G. Perrier. A procedure for automatic proof nets construction. In *LPAR'92, International Conference on Logic Programming and Automated Reasoning, LNAI 624*, pages 42–53, St. Petersburg, Russia, July 1992.

[12] J.Y. Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–102, 1987.

[13] J.Y. Girard. Towards a geometry of interaction. In J.W. Gray and A. Scedrov, editors, *Conference on Categories in Computer Science and Logic*, pages 69–108, Boulder, Colorado, June 1987.

[14] J. Harland and D. Pym. The uniform proof-theoretic foundation of linear logic programming. In *International Symposium on Logic Programming, MIT Press*, pages 304–318, San Diego, October 1991.

[15] J. Harland and D. Pym. On resolution in fragments of classical linear logic. In *LPAR'92, International Conference on Logic Programming and Automated Reasoning, LNAI 624*, pages 30–41, St. Petersburg, Russia, July 1992.

[16] J.S. Hodas and D. Miller. Logic programming in a fragment of intuitionistic linear logic. In *6th IEEE Symposium on Logic in Computer Science*, pages 32–42, Amsterdam, The Netherlands, July 1991.

[17] M.I. Kanovich. Horn programming in linear logic is NP-complete. In *7th IEEE Symposium on Logic in Computer Science*, pages 200–210, Santa-Cruz, California, June 1992.

[18] N. Kobayashi and A. Yonezawa. ACL - a concurrent linear logic programming paradigm. In *Int. Symposium on Logic Programming*, pages 279–294, Vancouver, October 1993.

[19] Y. Lafont. Interaction nets. In *17th ACM Symposium on Principles of Programming Languages*, pages 95–108, San Francisco, January 1990.

[20] P. Lincoln and J. Mitchell. Operational aspects of linear lambda calculus. In *7th IEEE Symposium on Logic in Computer Science*, pages 235–246, Santa-Cruz, California, June 1992.

[21] P. Lincoln, J. Mitchell, A. Scedrov, and N. Schankar. Decision problems for propositional linear logic. In *31st IEEE Symposium on Foundations of Computer Science*, pages 662–671, St. Louis, Missouri, October 1990.

[22] M. Masseron, C. Tollu, and J. Vauzeilles. Generating plans in linear logic. In *Foundations of Software Technology and Theoretical Computer Science, LNCS 472*, pages 63–75, Bangalore, India, December 1990.

[23] M. Masseron, C. Tollu, and J. Vauzeilles. Generating plans in linear logic I: Actions as proofs. *Theoretical Computer Science*, 113(2):349–371, 1993.

[24] J. Meseguer and N. Marti-Oliet. From petri nets to linear logic. In *Category Theory and Computer Science, LNCS 389*, pages 313–340, Manchester, September 1989.

[25] J. Mitchell. Introduction to linear logic. *Sigact News*, 23(2):29–37, 1992.

[26] N. Shankar. Proof search in the intuitionistic sequent calculus. In *11th Conference on Automated DEduction, LNAI 607*, pages 522–536, Saratoga Springs, June 1992.

[27] T. Tammet. Proof search strategies in linear logic. Programming Methodology Group Report 70, Chalmers University Group, University of Göteborg, 1993.

[28] A.S. Troelstra. *Lectures on Linear Logic*, volume 29 of *Lecture Notes*. CSLI, 1992.

[29] L.A. Wallen. *Automated Proof search in Non-Classical Logics*. MIT Press, 1990.

A Proof of permutability theorem

Case by case analysis according to the partition of the inference rules into two groups: those depending on the context, i.e, of type $\&, !, \forall$ and the others.

1) $\underline{type(I_2) = \&}$.

To have I_1 permutable with I_2, we need in Π an inference J_1 of the same type as I_1 and the conclusion of which coincides with the premiss of I_2 that is not the conclusion of I_1. For any type for I_1, we can find a counter-example that does not verify this condition. Then I_1 is not always permutable with I_2.

2) $\underline{type(I_2) = !}$.

If I_1 is in permutation position with I_2 the context of I_2, being under the form $?\Delta$, contains $\Delta_p(I_1)$ and then we have two possibilities:

α) $\Delta_p(I_1) = \emptyset$ and $type(I_1) = \ cut$,

β) $\Delta_p(I_1) = \{?A\}$ and $type(I_1) \in \{?, w?, c?\}$.

2.1) $\underline{type(I_1) = cut}$.

In this case, Π has the following form

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash F, G, ?\Delta_n^1\end{array} \quad \begin{array}{c}\vdots\\ \vdash F^\perp, ?\Delta_n^2\end{array}}{\vdash G, ?\Delta_n^1, ?\Delta_n^2}\ cut}{\vdash !G, ?\Delta_n^1, ?\Delta_n^2}\ !$$

Let us consider the proof

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash F, G, ?\Delta_n^1\end{array}}{\vdash F, !G, ?\Delta_n^1}\ ! \quad \dfrac{\vdots}{\vdash F^\perp, ?\Delta_n^2}}{\vdash !G, ?\Delta_n^1, ?\Delta_n^2}\ cut$$

If F does not have the form ?F then this proof is not correct and I_1 is not always permutable with I_2. Let us remark that if we extend the notion of inference permutability by adding the possibility of inserting new inferences in the resulting proof as in [10] then we can say that I_1 is permutable with I_2 in a certain sense.

In this permutation, we add two inferences ? and ! to obtain the following proof

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash F, G, ?\Delta_n^1\end{array}}{\vdash ?F, G, ?\Delta_n^1}\ ?}{\vdash ?F, !G, ?\Delta_n^1}\ ! \quad \dfrac{\begin{array}{c}\vdots\\ \vdash F^\perp, ?\Delta_n^2\end{array}}{\vdash !F^\perp, ?\Delta_n^2}\ !}{\vdash !G, ?\Delta_n^1, ?\Delta_n^2}\ cut$$

2.2) $\underline{type(I_1) = ?}$.

The following counter-example shows that in general I_1 is not permutable with I_2.

$$\dfrac{\dfrac{\overline{\vdash A, A^\perp}}{\vdash ?A, A^\perp}\ ?}{\vdash ?A, !A^\perp}\ !$$

2.3) $\underline{type(I_1) = w?}$.

In this case, we have the transformation

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash G, ?\Delta_n\end{array}}{\vdash ?F, G, ?\Delta_n}\,w?}{\vdash ?F, !G, ?\Delta_n}\,! \quad\leadsto\quad \dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash G, ?\Delta_n\end{array}}{\vdash !G, ?\Delta_n}\,!}{\vdash ?F, !G, ?\Delta_n}\,w?$$

that is correct. Then I_1 is permutable with I_2.

2.4) $\underline{type(I_1) = c?}$.
In this case, we have the transformation

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash ?F, ?F, G, ?\Delta_n\end{array}}{\vdash ?F, G, ?\Delta_n}\,c?}{\vdash ?F, !G, ?\Delta_n}\,! \quad\leadsto\quad \dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash ?F, ?F, G, ?\Delta_n\end{array}}{\vdash ?F, ?F, !G, ?\Delta_n}\,!}{\vdash ?F, !G, ?\Delta_n}\,c?$$

that is correct. Then I_1 is permutable with I_2.
3) $\underline{type(I_2) = \forall}$.
Necessarily, the variable y quantified by I_2 is not free in Δ_p. If I_1 is not of type cut, $\forall$ or $\exists$ the variable y is not free also in Δ_a (or Δ_a^1 and Δ_a^2). Moreover, if I_1 is independent of the context then I_1 is permutable with I_2. Let us analyze the cases with potential problems.
3.1) $\underline{type(I_1) = cut}$.
The following counter-example shows that I_1 is not in general permutable with I_2.

$$\dfrac{\dfrac{\dfrac{}{\vdash a(y), a(y)^{\perp}} \qquad \dfrac{\dfrac{}{\vdash a(y), a(y)^{\perp}}}{\vdash \exists x a(x), a(y)^{\perp}}\,\exists}{\vdash a(y)^{\perp}, \exists x a(x)}\,cut}{\vdash \forall x a(x)^{\perp}, \exists x a(x)}\,\forall$$

3.2) $\underline{type(I_1) = \&}$.
In this case, we have the transformation

$$\dfrac{\dfrac{\vdash F, H[y/x], \Delta_n \quad \vdash G, H[y/x], \Delta_n}{\vdash F\&G, H[y/x], \Delta_n}\,\&}{\vdash F\&G, \forall x H, \Delta_n}\,\forall \quad\leadsto\quad \dfrac{\dfrac{\vdash F, H[y/x], \Delta_n}{\vdash F, \forall x H, \Delta_n}\,\forall \quad \dfrac{\vdash G, H[y/x], \Delta_n}{\vdash G, \forall x H, \Delta_n}\,\forall}{\vdash F\&G, \forall x H, \Delta_n}\,\&$$

that is a correct proof. Then I_1 is permutable with I_2.
3.3) $\underline{type(I_1) = \forall}$.
In this case, we have the transformation

$$\dfrac{\dfrac{\vdash F[u/x], G[z/y], \Delta_n}{\vdash \forall x F, G[z/y], \Delta_n}\,\forall}{\vdash \forall x F, \forall y G, \Delta_n}\,\forall \quad\leadsto\quad \dfrac{\dfrac{\vdash F[u/x], G[z/y], \Delta_n}{\vdash F[u/x], \forall y G, \Delta_n}\,\forall}{\vdash \forall x F, \forall y G, \Delta_n}\,\forall$$

If the left-hand side proof Π is correct, let us show that the resulting proof Π' on the right-hand side is also correct. To do that, we have to prove that the two $\forall$ inferences of Π' are correct. Let us begin with the second one. The first $\forall$ inference of Π being correct, u is not free in $G[z/y], \Delta_n$. Then, a fortiori, u is not free in $\forall y G, \Delta_n$.
Let us examine now the first one. The second $\forall$ inference of Π being correct, z is not free in $\forall x F, \Delta_n$. The only case in which z could be free in $F[u/x], \Delta_n$ is the one where z is identical to

u. Then $G[z/y] \equiv G[u/y]$. Since u is not free in $G[z/y]$, i.e in $G[u/y]$, then $G[z/y] \equiv G$ and y is not free in G.
Thus, by association of any variable, that is neither free in G nor in $F[u/x]$, with the first $\forall$ inference of Π' we obtain a correct inference. In conclusion, Π' is correct and I_1 is permutable with I_2.
3.4) $\underline{type(I_1) = \exists}$.
The following counter-example shows that I_1 is not in general permutable with I_2.

$$\dfrac{\dfrac{\overline{\vdash a(y), a(y)^\perp}}{\vdash a(y), \exists x a(x)^\perp}\,\exists}{\vdash \forall x a(x), \exists x a(x)^\perp}\,\forall$$

3.5) $\underline{type(I_1) = !}$.
The following counter-example shows that I_1 is not in general permutable with I_2.

$$\dfrac{\dfrac{\dfrac{\overline{\vdash a, a^\perp}}{\vdash ?a, a^\perp}\,?}{\vdash ?a, !a^\perp}\,!}{\vdash \forall x(?a), !a^\perp}\,\forall$$

4) $\underline{type(I_1) = ! \text{ and } type(I_2) \notin \{\&, !, \forall\}}$.
4.1) $\underline{I_2 \text{ has one premise}}$.
Π has the following form

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash F, ?\Delta'_a, ?\Delta_n\end{array}}{\vdash !F, ?\Delta'_a, ?\Delta_n}\,!}{\vdash !F, \Delta'_p, ?\Delta_n}\,I_2$$

Let us consider the following proof

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash F, ?\Delta'_a, ?\Delta_n\end{array}}{\vdash F, \Delta'_p, ?\Delta_n}\,I'_2}{\vdash !F, \Delta'_p, ?\Delta_n}\,!.$$

It is correct iff Δ'_p is under the form $?\Delta$, i.e, iff I_2 is of type $?$, $w?$ ou $c?$. In these three cases, I_1 is always permutable with I_2 but not in the other cases.
4.2) $\underline{I_2 \text{ has two premises}}$.
Then I_2 is of the *cut* or $\otimes$ type and Π has the form

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash F, ?\Delta'^1_a, ?\Delta_n\end{array}}{\vdash !F, ?\Delta'^1_a, ?\Delta_n}\,! \qquad \dfrac{\vdots}{\vdash \Delta'^2_a, \Delta'_n}}{\vdash !F, \Delta'_p, ?\Delta_n, \Delta'_n}\,I_2$$

Δ'_n not being necessarily of the form $?\Delta$, I_1 is not always permutable with I_2.
5) $\underline{type(I_1) = \forall \text{ and } type(I_2) \notin \{\&, !, \forall\}}$.
5.1) $\underline{I_2 \text{ has one premise}}$.
In this case, Π has the following form

$$\dfrac{\dfrac{\Pi_1\left\{\begin{array}{c}\vdots\\ \vdash F[y/x],\Delta'_a,\Delta_n\end{array}\right.}{\vdash \forall xF,\Delta'_a,\Delta_n}\,\forall}{\vdash \forall xF,\Delta'_p,\Delta_n}\,I_2$$

Let z be a variable that is not free in all intermediate conclusions of Π over I_2 included.

$$\dfrac{\dfrac{\Pi_1[z/y]\left\{\begin{array}{c}\vdots\\ \vdash F[z/x],\Delta'_a,\Delta_n\end{array}\right.}{\vdash F[z/x],\Delta'_p,\Delta_n}\,I'_2}{\vdash \forall xF,\Delta'_p,\Delta_n}\,\forall$$

It is correct because I_2 is of one type that does not depend on the context and z is not free in Δ'_p,Δ_n. Then I_1 is always permutable with I_2 (when I_2 is not of type $\oplus$ or $w?$ a renaming is not necessary).

5.2) $\underline{I_2 \text{ has two premises}}$.

I_2 is then of the *cut* or $\otimes$ type and Π has the form

$$\dfrac{\dfrac{\Pi_1\left\{\begin{array}{c}\vdots\\ \vdash F[y/x],\Delta'^1_a,\Delta_n\end{array}\right.}{\vdash \forall xF,\Delta'^1_a,\Delta_n}\,\forall \qquad \Pi_2\left\{\begin{array}{c}\vdots\\ \vdash \Delta'^2_a,\Delta'_n\end{array}\right.}{\vdash \forall xF,\Delta'_p,\Delta_n,\Delta'_n}\,I'_2$$

Let z be a variable that is not free in all intermediate conclusions of Π over I_2 included.

$$\dfrac{\dfrac{\Pi_1[z/y]\left\{\begin{array}{c}\vdots\\ \vdash F[z/x],\Delta'^1_a,\Delta_n\end{array}\right. \qquad \Pi_1[z/y]\left\{\begin{array}{c}\vdots\\ \vdash \Delta'^2_a\Delta'_n\end{array}\right.}{\vdash F[z/x],\Delta'_p,\Delta_n,\Delta'_n}\,I'_2}{\vdash \forall xF,\Delta'_p,\Delta_n,\Delta'_n}\,\forall$$

Here y is not free in Δ'^1_a and Δ'^2_a.

The proof is correct and then I_1 is always permutable with I_2.

6) $\underline{type(I_1) = \& \text{ and } type(I_2) \notin \{\&,\forall\}}$.

6.1) $\underline{I_2 \text{ has one premise}}$.

In this case, Π has the following form

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash F,\Delta'_a,\Delta_n\end{array} \qquad \begin{array}{c}\vdots\\ \vdash G,\Delta'_a,\Delta_n\end{array}}{\vdash F\&G,\Delta'_a,\Delta_n}\,\&}{\vdash F\&G,\Delta'_p,\Delta_n}\,I_2$$

Let us consider the proof

$$\dfrac{\dfrac{\vdash F,\Delta'_a,\Delta_n}{\vdash F,\Delta'_p,\Delta_n}\,I_2 \qquad \dfrac{\vdash G,\Delta'_a,\Delta_n}{\vdash G,\Delta'_p,\Delta_n}\,I_2}{\vdash F\&G,\Delta'_p,\Delta_n}\,\&$$

It is correct because I_2 does not depend on the context and then I_1 is always permutable with I_2.

6.2) $\underline{I_2 \text{ has two premises}}$.

I_2 is then of the *cut* or $\otimes$ type and then Π has the form

$$\cfrac{\cfrac{\begin{array}{cc} \vdots & \vdots \\ \vdash F, \Delta_a'^1, \Delta_n & \vdash G, \Delta_a'^1, \Delta_n \end{array}}{\vdash F \& G, \Delta_a'^1, \Delta_n}\ \& \qquad \begin{array}{c} \vdots \\ \vdash \Delta_a'^2, \Delta_n' \end{array}}{\vdash F \& G, \Delta_p', \Delta_n, \Delta_n'}\ I_2$$

Let us consider the proof

$$\cfrac{\cfrac{\vdash F, \Delta_a'^1, \Delta_n \quad \vdash \Delta_a'^2, \Delta_n'}{\vdash F, \Delta_p', \Delta_n, \Delta_n'}\ I_2' \quad \cfrac{\vdash G, \Delta_a'^1, \Delta_n \quad \vdash \Delta_a'^2, \Delta_n'}{\vdash G, \Delta_p', \Delta_n, \Delta_n'}\ I_2'}{\vdash F \& G, \Delta_p', \Delta_n, \Delta_n'}\ \&$$

It is correct and then I_1 is always permutable with I_2.
7) $\underline{type(I_i)_{i \in \{1,2\}} \notin \{\&, !, \forall\}}$.
7.1) I_1 and I_2 have only one premise.
Π has the following form

$$\cfrac{\cfrac{\begin{array}{c} \vdots \\ \vdash \Delta_a, \Delta_a', \Delta_n \end{array}}{\vdash \Delta_p, \Delta_a', \Delta_n}\ I_1}{\vdash \Delta_p, \Delta_p', \Delta_n}\ I_2$$

The following proof is correct (because I_1 and I_2 do not depend on the context).

$$\cfrac{\cfrac{\begin{array}{c} \vdots \\ \vdash \Delta_a, \Delta_a', \Delta_n \end{array}}{\vdash \Delta_a, \Delta_p', \Delta_n}\ I_2'}{\vdash \Delta_p, \Delta_p', \Delta_n}\ I_1'$$

Then I_1 is always permutable with I_2.
7.2) I_1 has one premise and I_2 has two premises.
I_2 is of the *cut* or $\otimes$ type and then Π has the following form:

$$\cfrac{\cfrac{\begin{array}{c} \vdots \\ \vdash \Delta_a, \Delta_a'^1, \Delta_n \end{array}}{\vdash \Delta_p, \Delta_a'^1, \Delta_n}\ I_1 \qquad \begin{array}{c} \vdots \\ \vdash \Delta_a^2, \Delta_n' \end{array}}{\vdash \Delta_p, \Delta_p', \Delta_n, \Delta_n'}\ I_2$$

The following proof is correct (because I_1 does not depend on the context).

$$\cfrac{\cfrac{\vdash \Delta_a, \Delta_a'^1, \Delta_n \quad \vdash \Delta_a'^2, \Delta_n'}{\vdash \Delta_a, \Delta_p', \Delta_n, \Delta_n'}\ I_2'}{\vdash \Delta_p, \Delta_p', \Delta_n, \Delta_n'}\ I_1'$$

Then I_1 is always permutable with I_2.
7.3) I_1 has two premises and I_2 has one premise.
a) $\underline{type(I_2) \notin \{c?, \wp\}}$.
In this case, Δ_a' has at most one formula and then cannot be *cut* into two non-empty sets: it is completely included in one of the premises of I_1 (for example, the left one).

$$\dfrac{\dfrac{\begin{matrix}\vdots\\ \vdash \Delta_a^1, \Delta'_a, \Delta_n\end{matrix} \qquad \begin{matrix}\vdots\\ \vdash \Delta_a^2, \Delta'_n\end{matrix}}{\vdash \Delta_p, \Delta'_a, \Delta_n, \Delta'_n}\, I_1}{\vdash \Delta_p, \Delta'_p, \Delta_n, \Delta'_n}\, I_2$$

The following proof is correct (because I_1 does not depend on the context).

$$\dfrac{\dfrac{\begin{matrix}\vdots\\ \vdash \Delta_a^1, \Delta'_a, \Delta_n\end{matrix}}{\vdash \Delta_a^1, \Delta'_p, \Delta_n}\, I'_2 \qquad \begin{matrix}\vdots\\ \vdash \Delta_a^2, \Delta'_n\end{matrix}}{\vdash \Delta_p, \Delta'_p, \Delta_n, \Delta'_n}\, I'_1$$

Then I_1 is always permutable with I_2.

b) $type(I_2) = \wp$.

The following counter-examples show that, in this case, I_1 is not always permutable with I_2.

$$\dfrac{\dfrac{\dfrac{}{\vdash A, A^\perp} \qquad \dfrac{}{\vdash B, B^\perp}}{\vdash A \otimes B, A^\perp, B^\perp}\, \otimes}{\vdash A \otimes B, A^\perp \wp B^\perp}\, \wp \qquad\qquad \dfrac{\dfrac{\dfrac{}{\vdash A, A^\perp} \qquad \dfrac{}{\vdash B, B^\perp}}{\vdash A, A^\perp}\, cut}{\vdash A \wp A^\perp}\, \wp$$

c) $type(I_2) = c?$.

The following counter-examples show that, in this case, I_1 is not always permutable with I_2.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash A, A^\perp}}{\vdash A, ?(A^\perp)}\, ? \qquad \dfrac{\dfrac{}{\vdash A, A^\perp}}{\vdash A, ?(A^\perp)}\, ?}{\vdash A \otimes A, ?(A^\perp), ?(A^\perp)}\, \otimes}{\vdash A \otimes A, ?(A^\perp)}\, c?$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash B, B^\perp} \qquad \dfrac{}{\vdash A, A^\perp}}{\vdash B^\perp \otimes A^\perp, B, A}\, \otimes}{\vdash B^\perp \otimes A^\perp, B, ?A}\, d? \qquad \dfrac{\dfrac{\dfrac{}{\vdash B, B^\perp} \qquad \dfrac{}{\vdash A, A^\perp}}{\vdash B \otimes A^\perp, B^\perp, A}\, \otimes}{\vdash B \otimes A^\perp, B^\perp, ?A}\, d?}{\vdash B^\perp \otimes A^\perp, B \otimes A^\perp, ?A, ?A}\, cut}{\vdash B^\perp \otimes A^\perp, B \otimes A^\perp, ?A}\, c?$$

7.4) I_1 and I_2 have two premises.

I_2 is of the *cut* or $\otimes$ type then Π has the following form:

$$\dfrac{\dfrac{\begin{matrix}\vdots\\ \vdash \Delta_a^1, \Delta_a'^1, \Delta_n\end{matrix} \qquad \begin{matrix}\vdots\\ \vdash \Delta_a^2, \Delta''_n\end{matrix}}{\vdash \Delta_p, \Delta_a'^1, \Delta_n, \Delta'_n}\, I_1 \qquad \begin{matrix}\vdots\\ \vdash \Delta_a^2, \Delta''_n\end{matrix}}{\vdash \Delta_p, \Delta'_p, \Delta_n, \Delta'_n, \Delta''_n}\, I_2.$$

The following proof is correct and then I_1 is always permutable with I_2.

$$\dfrac{\dfrac{\vdash \Delta_a^1, \Delta_a'^1, \Delta_n \qquad \vdash \Delta_a'^2, \Delta''_n}{\vdash \Delta_a^1, \Delta'_p, \Delta_n, \Delta''_n}\, I'_2 \qquad \vdash \Delta_a^2, \Delta''_n}{\vdash \Delta_p, \Delta'_p, \Delta_n, \Delta'_n, \Delta''_n}\, I'_1.$$

B Proof of theorem 5.1 (complexity reduction)

Theorem B.1 *(complexity reduction)*
Let Π be a proof in LL of a sequent $\vdash \Delta$ with only one cut, there exists a proof Π' of $\vdash \Delta$ such that $c(\Pi') < c(\Pi)$.

Proof B.1 *Let I be the unique cut in Π. By theorem 4.3, there exists a proof Π'', obtained by backward movement of I in Π, such that for any inference I' resulting from this movement, I' is immediately preceded by two inferences introducing the active formulae of I'.*
Thus, from definition 5.3, we have c(I') which is equal to c(I). In fact, the problem is to reduce the complexity of each cut I' considering the subproof Π_1 of Π'' ending with I'.
Hence, Π_1 has only one cut I' and then $c(\Pi_1) = c(I') = c(\Pi)$.
Let us consider such a cut I', with several cases according to the form of the active formulae of I'.

1) The active formulae are atomic or constant.
Π_1 can have three different forms:

$$a)\ \frac{\dfrac{}{\vdash \top, \Delta} \qquad \dfrac{}{\vdash 0, \top, \Delta'}}{\vdash \top, \Delta, \Delta'} \qquad\qquad b)\ \frac{\dfrac{}{\vdash 1} \qquad \dfrac{\begin{array}{c}\vdots\\ \vdash \Delta\end{array}\ \}\Pi_2}{\vdash \bot, \Delta}}{\vdash \Delta} \qquad\qquad c)\ \frac{\dfrac{}{\vdash A, A^{\perp}} \qquad \dfrac{}{\vdash A, A^{\perp}}}{\vdash A, A^{\perp}}$$

In the case a), it can be replaced by the proof $\Pi'_1 \equiv \dfrac{}{\vdash \top, \Delta, \Delta'}$ *with* $c(\Pi'_1) = 0 < c(\Pi_1)$. *In the case b), it can be replaced by the proof Π_2 which contains no more cut and then $c(\Pi_2) = 0 < c(\Pi_1)$. Finally, in case c), it can be replaced by the proof* $\Pi'_1 \equiv \dfrac{}{\vdash A, A^{\perp}}$ *which contains no more cut and then $c(\Pi'_1) = 0 < c(\Pi_1)$.*

2) The active formulae are of the $\otimes$ and $\wp$ type.
In this case, Π_1 has the form

$$\frac{\dfrac{\begin{array}{c}\vdots\\ \vdash F, \Delta_n\end{array} \qquad \begin{array}{c}\vdots\\ \vdash G, \Delta'_n\end{array}}{\vdash F \otimes G, \Delta_n, \Delta'_n}\otimes \qquad \dfrac{\begin{array}{c}\vdots\\ \vdash F^{\perp}, G^{\perp}, \Delta''_n\end{array}}{\vdash F^{\perp}\wp G^{\perp}, \Delta''_n}\wp}{\vdash \Delta_n, \Delta'_n, \Delta''_n}\ cut$$

and we can replace it by the following proof Π'_1

$$\frac{\dfrac{\begin{array}{c}\vdots\\ \vdash F, \Delta_n\end{array} \qquad \begin{array}{c}\vdots\\ \vdash F^{\perp}, G^{\perp}, \Delta''_n\end{array}}{\vdash G^{\perp}, \Delta_n, \Delta'_n}\ cut \qquad \begin{array}{c}\vdots\\ \vdash G, \Delta'_n\end{array}}{\vdash \Delta_n, \Delta'_n, \Delta''_n}\ cut$$

and we have $c(\Pi'_1) = c(F) + c(F^{\perp})$ and $c(\Pi_1) = c(F \otimes G) + c(F^{\perp}\wp G^{\perp}) = sup\{c(F), c(G)\} + sup\{c(F^{\perp}), c(G^{\perp})\}$ +2. Then $c(\Pi'_1) < c(\Pi_1)$.

3) The active formulae are of the $\oplus$ and & type.
In this case, Π_1 has the form

$$\dfrac{\dfrac{\vdots \atop \vdash F, \Delta_n \qquad \vdots \atop \vdash G, \Delta_n}{\vdash F\&G, \Delta_n}\,\& \qquad \dfrac{\vdots \atop \vdash F^{\perp}, \Delta'_n}{\vdash F^{\perp} \oplus G^{\perp}, \Delta'_n}\,\oplus}{\vdash \Delta_n, \Delta'_n}\; cut$$

and we can replace it by the following proof Π'_1

$$\dfrac{\vdots \atop \vdash F, \Delta_n \qquad \vdots \atop \vdash F^{\perp}, \Delta'_n}{\vdash \Delta_n, \Delta'_n}\; cut$$

and we have $c(\Pi'_1) = c(F) + c(F^{\perp})$ *and* $c(\Pi_1) = c(F\&G) + c(F^{\perp} \oplus G^{\perp}) = sup\{c(F), c(G)\} + sup\{c(F^{\perp}), c(G^{\perp})\} + 2$*. Then* $c(\Pi'_1) < c(\Pi_1)$*.*

4) <u>The active formulae are of the $\forall$ and $\exists$ type.</u>
In this case, Π_1 *has the form*

$$\dfrac{\dfrac{\Pi_2 \left\{ \vdots \atop \vdash F[y/x], \Delta_1 \right.}{\vdash \forall x F, \Delta_1}\, w? \qquad \dfrac{\vdots \atop \vdash F^{\perp}[t/x], \Delta_2}{\vdash \exists x F^{\perp}, \Delta_2}\, !}{\vdash \Delta_1, ?\Delta_2}\; cut$$

and we can replace it by the following proof Π'_1

$$\dfrac{\Pi'_2 \left\{ \vdots \atop \vdash F[t/x], \Delta_1 \right. \qquad \vdash F^{\perp}[t/x], \Delta_2}{\vdash \Delta_1, \Delta_2}$$

when Π'_2 *is obtained from* Π_2 *by substitution of* $F[t/x]$ *to* $F[y/x]$ *with a renaming of the variables that do not respect the condition on* $\forall$ *rule. Thus we have* $c(\Pi'_1) = c(F[t/x]) + c(F^{\perp}[t/x])$ *and* $c(\Pi_1) = c(F[t/x]) + c(F^{\perp}[t/x] + 2)$ *and then* $c(\Pi'_1) < c(\Pi_1)$*.*

5) <u>The active formulae are of the ! and ? type.</u>
In this case, we have three subcases depending on the way the active formula of type ? has been introduced.
5.1) <u>Introduction by an inference of the ? type.</u>
In this case, Π_1 *has the form*

$$\dfrac{\dfrac{\vdots \atop \vdash F, \Delta_n}{\vdash ?F, \Delta_n}\, ? \qquad \dfrac{\vdots \atop \vdash F^{\perp}, ?\Delta'_n}{\vdash !F^{\perp}, ?\Delta'_n}\, !}{\vdash \Delta_n, ?\Delta'_n}\; cut$$

and we can replace it by the following proof Π'_1

$$\dfrac{\vdots \atop \vdash F, \Delta_n \qquad \vdots \atop \vdash F^{\perp}, ?\Delta'_n}{\vdash \Delta_n, ?\Delta'_n}\; cut$$

and we have $c(\Pi'_1) = c(F) + c(F^\perp)$ *and* $c(\Pi_1) = c(F) + c(F^\perp) + 2$. *Then* $c(\Pi'_1) < c(\Pi_1)$.

5.2) *Introduction by an inference of the w? type.*

In this case, Π_1 *has the form*

$$\dfrac{\dfrac{\Pi_2\left\{\begin{array}{c}\vdots\\ \vdash \Delta_n\end{array}\right.}{\vdash ?F, \Delta_n}\;w? \qquad \dfrac{\begin{array}{c}\vdots\\ \vdash F^\perp, ?\Delta'_n\end{array}}{\vdash !F^\perp, ?\Delta'_n}\;!}{\vdash \Delta_n, ?\Delta'_n}\;cut$$

and we can replace it by the following proof Π'_1

$$\begin{array}{c}\dfrac{\Pi_2\left\{\begin{array}{c}\vdots\\ \vdash \Delta_n\end{array}\right.}{\vdots}\\ \vdash \Delta_n, ?\Delta'_n\end{array}$$

and we have $c(\Pi'_1) = 0 < c(\Pi_1)$.

5.3) *Introduction by an inference of the c? type.*

In this case, Π_1 *has the form*

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash ?F, ?F, \Delta_n\end{array}}{\vdash ?F, \Delta_n}\;c? \qquad \dfrac{\begin{array}{c}\vdots\\ \vdash F^\perp, ?\Delta'_n\end{array}}{\vdash !F^\perp, ?\Delta'_n}\;!}{\vdash \Delta_n, ?\Delta'_n}\;cut$$

and we can replace it by the following proof Π'_1

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash ?F, ?F, \Delta_n\end{array} \qquad \begin{array}{c}\vdots\\ \vdash !F^\perp, ?\Delta'_n\end{array}}{\vdash ?F, \Delta_n, ?\Delta'_n}\;cut \qquad \begin{array}{c}\vdots\\ \vdash !F^\perp, ?\Delta'_n\end{array}}{\vdash \Delta_n, ?\Delta'_n, ?\Delta'_n}\;cut}{\begin{array}{c}\vdots\\ \vdash \Delta_n, ?\Delta'_n\end{array}}\;c?$$

and we have $c(\Pi'_1) = c(?F) + c(!F^\perp)$ *and* $c(\Pi_1) = 1 + c(\Pi'_1)$. *Then* $c(\Pi'_1) < c(\Pi_1)$.