Constraint Based Proof Construction in Non-Commutative Logic

Jean-Marc Andreoli, Roberto Maieli and Paul Ruet

CNRS - Institut de Mathématiques de Luminy
163 avenue de Luminy, Case 907, 13288 Marseille Cedex 9 (France)
andreoli,maieli,ruet@iml.univ-mrs.fr

July 13, 2001

Abstract

This work presents a computational interpretation of the construction process for cyclic (CyLL) and non-commutative (NL) sequential proofs. We assume a proof construction paradigm, based on a normalization procedure, known as focusing which manages efficiently the non-determinism of the construction.

Similarly to the linear case, a new formulation of focusing for NL is used to introduce a general constraint-based technique in order to deal with partial information during proof construction. In particular, the procedure develops through construction steps building “intermediate objects” called abstract proofs, similar to the design of indices.

1 Introduction

We are interested here in the computational paradigm of proof construction in logical sequent calculi. The most straightforward (and naive) proof construction algorithm starts with a single open node labeled by a given sequent (or even a single formula), then tries to incrementally construct a proof by repeatedly expanding each open node, selecting and applying an inference from the sequent calculus, thus possibly introducing new open nodes. This provides an interesting computational model, particularly adapted to capture non-deterministic processes, since proof construction itself is intrinsically non-deterministic: in the naive procedure, for example, many choices of different kinds (choice of the principal formula, choice of the inference) have to be made at each expansion step.

However, it is well-known that, due to intrinsic permuability properties of the inferences in sequent calculi, some strategy is needed in order to avoid making sets of choices which lead to the same object (modulo permutation of inferences). Such a strategy, called focusing, has been proposed in [2]. It is based on the generic concept of polarity of formulas, and therefore applies to any logical system where connectives have polarities, such as logic logic or non-commutative logic [1, 14]. Focusing deals with two important forms of irrelevant non-determinism in proof construction: on one hand, the instant of the decomposition of a negative connective is simply irrelevant; on the other hand, the interval between two decompositions of positive connectives, if they belong to the same formula and one is an immediate successor of the other in the formula, is irrelevant. The strategy to avoid these forms of irrelevant non-determinism can be expressed in the sequent calculi themselves by modifying the syntax of the sequents, with the introduction of a distinguished formula called the “focus”. Such focusing sequent calculi have thus been proposed in various contexts for linear logic [9] and in a non-commutative framework [12, 13]. There is also an alternative presentation which does not rely on syntactic conventions while capturing exactly the same content: keep sequents as simple as possible (eg. they are made of atoms only), but refine the inferences of the calculus themselves. Such a refined, simplified calculus, called the (focusing) bipolar calculus, has been presented in [3] for linear logic. It is strictly isomorphic to the focusing sequent calculus of linear logic, so proof construction can be performed equivalently in both systems. We extend here the focusing bipolar calculus to non-commutative logic, without problem given the genericity of the approach.

Now, the naive proof construction procedure can be directly applied to the focusing bipolar calculus, but this is still unsatisfactory. Indeed, although focusing eliminates a lot of irrelevant non-determinism, there still remains a source of non-determinism which are untractable in the naive approach. The most well-known one appears in the first-order case with the existential quantification rule [10]: a choice of a term to assign to the quantified variable has to be made, but there are infinitely many possibilities. A random enumeration is clearly unsuitable, essentially because two different random choices may lead to essentially the same object, differing only by a term which, anyway, is irrelevant to the proof. For example, here are two proofs of the formula $\forall x \ p(x) \rightarrow \exists x \ p(x)$ which differ by a purely irrelevant choice (of the term to assign to $x$):

...
Again, we need a strategy to avoid such irrelevant non-determinism. In the case of the existential quantification, the solution is quite familiar: it is based on unification, which performs the choice of the term for an existentially quantified variable in a lazy way, making actual choices only when they are needed somewhere in the proof. This leads the construction procedure to manipulate not ground proofs, but abstract proofs containing only partial information about the object being constructed (partiality derives from the presence of uninstantiated variables occurring in the terms). This also means that, at each step, some constraints on the different variables have to be propagated across the proof, when variables are shared between multiple nodes: this is the role of the unification process. We thus obtain a slightly less naive construction process which interleaves the usual naive expansion process and the “dual” unification process which works in the reverse direction.

There is another case of irrelevant non-determinism which the naive procedure cannot avoid, namely caused by the positive multiplicative connectives: to decompose such a connective, a choice of splitting of the context has to be performed. If we start the construction with a single formula, this choice always involves only finite alternatives. But if we start with a sequent which is itself partially defined (not only the atoms it contains may be partially instantiated, but they may not all be known), the number of alternatives may become infinite. And in both cases, as before, a random enumeration is inadequate. A solution based on lazy splitting, and capable of dealing with partially defined (or even, possibly, completely undefined) sequents, has been proposed, for the focusing linear logic in [3]. It couples the naive construction procedure with a non-deterministic but finite constraint propagation procedure, thus leading to a constraint based construction mechanism. We study here the extension of this mechanism to non-commutative logic. Interestingly, it is shown that it extends straightforwardly to cyclic logic [1, 5], but not directly to non-commutative logic, although it is an extension of both linear and cyclic logic. We then propose a new solution, adapted to the full non-commutative case.

The overall goal of the line of research which this paper contributes to is two-fold: on the one hand, define a new proof construction mechanism capable of dealing with partial information, on the other hand eliminate sources of irrelevant non-determinism in the construction process. Both aspects are important in modeling real situations, esp. in the context of widely distributed applications, such as Internet applications, where there is no central focus with a vision of the whole execution (hence the importance of dealing with partial information), and programs are strongly influenced by their environment, which, at any reasonable level of abstraction, behaves in a highly non-deterministic way. The proof construction paradigm provides an appropriate level of abstraction to understand coordination issues in such applications, as shown by the CLF system [4], a component-based middleware infrastructure built around the central notion of resource (linear logic) and coordination viewed as resource manipulation (proof construction). However, CLF exploits only a very limited fragment of linear logic. The constraint based proof construction mechanism, developed here in the non-commutative case, offers perspectives for further extensions of the CLF platform in particular, and for a better understanding of coordination issues in distributed applications in general.

2 Non-commutative Logic

2.1 Order varieties

An order variety [1, 14] on a given set \( X \) is a ternary relation \( \alpha \) which is:

\[\forall x, y, z \in X, \alpha(x, y, z) \Rightarrow \alpha(y, z, x) \quad \text{cyclic}\]
\[\forall x, y \in X, \neg\alpha(x, y, y) \quad \text{anti-reflexive}\]
\[\forall x, y, z, t \in X, \alpha(x, y, z) \text{ and } \alpha(z, t, x) \Rightarrow \alpha(y, z, t) \quad \text{transitive}\]
\[\forall x, y, z, t \in X, \alpha(x, y, z) \Rightarrow \alpha(t, y, z) \text{ or } \alpha(x, t, z) \text{ or } \alpha(x, y, t) \quad \text{spreading}\]

For instance, any oriented cycle \( G = \{a_1 \to \cdots \to a_n \to a_1\} \) induces a total order variety \( r(G) \) on the set of its vertices by: \( r(G)(x, y) \iff y \text{ is between } x \text{ and } z \text{ in } G \); this order variety is denoted by \( a_1a_2 \cdots a_n = a_2 \cdots a_n a_1 \), etc. Given a set \( X \) and a partial order \( \omega \), we denote by \( X\omega \) or \( \omega X \) the cyclic closure of \( X \times \omega \), which is clearly an order variety on \( X \uplus \omega \). The spreading condition enables to systematically give “presentations” of order varieties as orders in a reversible way, whence the name. The correspondence is as follows.

Given an order variety \( \alpha \) on \( X \) and \( x \in X \), we may define an order \( \alpha_x \) on \( X \setminus \{x\} \) by:

\[\{\alpha_x(y, z) \iff \alpha(x, y, z)\]
Conversely, given a partial order \( \omega \) and \( z \in X \), let \( \omega(x, y | z) \) denote the following ternary relation:

\[
\omega(x, y | z) \iff (\omega(x, z) \land \omega(y, z)) \\
\land (\omega(z, x) \land \omega(z, y))
\]

expressing that \( z \) is in the same relation with \( x \) and \( y \) in \( \omega \); then we may define an order variety \( \mathcal{V} \) on \( X \), the closure of \( \omega \), by:

\[
\mathcal{V}(x, y) \quad \text{and} \quad \omega(x, y | z) \quad \text{or} \\
\omega(z, x) \quad \text{and} \quad \omega(z, x | y)
\]

It is shown in [1] that \( \mathcal{V} \) is indeed an order variety. When \( \mathcal{V} = \alpha \), we say that \( \omega \) presents \( \alpha \). Given two orders \( \omega \) and \( \tau \), we may define the following orders on the disjoint union \( |\omega| + |\tau| \) of their supports:

\[
\omega < \tau = \omega + \tau + |\omega| \times |\tau| \\
\omega \parallel \tau = \omega + \tau
\]

One proves easily that the closure identifies series and parallel sums:

\[
\mathcal{V}(x, y) = \mathcal{V}(\omega) \land \mathcal{V}(\tau)
\]

The above order variety is denoted \( \omega * \tau \) and called the gluing of \( \omega \) and \( \tau \). It enjoys:

\[
\omega * \tau = \mathcal{V}(\omega) \parallel \mathcal{V}(\tau) = |\omega| \times |\tau|
\]

The two processes of fixing a point in an order variety and gluing orders are related by the following equations:

\[
(\alpha)_x * x = \alpha \quad \text{and} \quad (\omega * x)_x = \omega
\]

for \( \alpha \) an order variety on a set \( X \), \( x \in X \) and \( \omega \) an order on \( X \setminus \{x\} \). These equations state that the species of order varieties in the sense of Joyal [7] has derivative the species of partial orders.

2.2 Series-parallel order varieties

Series-parallel orders are those obtained from the unique orders on singletons (the empty relation \( \emptyset \) with support \( \{x\} \)) by series and parallel sums. For a more substantial survey, see [11].

Series-parallel order varieties are precisely those order varieties which can be presented by a series-parallel order. A series-parallel order variety \( \alpha \) on a set \( X \) can be represented by a rooted planar tree (or seaweed, or salg) with leaves labeled by elements of \( X \) and ternary nodes labeled by \( <, > \) or \( || \): take an arbitrary presentation of \( \alpha \) as a series-parallel order \( \omega \), write \( \omega \) as a non-unique (associativity, commutativity) — planar binary tree \( t \) with leaves labeled by elements of \( X \), and root and nodes labeled by \( < \) or \( || \) for series and parallel sum respectively; then remove the root of \( t \).

For instance \( (x < y < z) || (t < u) \) can be represented by:
To read the seaweed, take three leaves \( a, b, c \) and let \( \varepsilon \) be the node at the intersection of the three paths \( \alpha, \beta \) and \( \gamma \); then \( (a, b, c) \) is in the order variety iff:

- the node \( \varepsilon \) is labeled by \( a \) and
- the paths \( \varepsilon a, \varepsilon b \) and \( \varepsilon c \) are in this cyclic order while moving clockwise around \( \varepsilon \).

Restrictions to a subset \( X \) are denoted \( \omega|_X, \alpha|_X \); restriction preserves clearly the structures of order and order variety, and preserves series-parallelism. From now on, we consider only series-parallel orders and order varieties.

2.3 Entropy

Entropy \( \sqsubseteq \) is the relation between series-parallel orders on the same given set defined by

\[
\omega \sqsubseteq \tau \quad \text{iff} \quad \omega \preceq \tau \quad \text{and} \quad \tau \preceq \omega
\]

Entropy is clearly a partial order, compatible with restriction and with the symmetric-series, series and parallel sums of orders. In the series-parallel case, \( \sqsubseteq \) is the least reflexive transitive relation between series-parallel orders on the same set such that:

\[
\omega \leq \omega_1 \leq \omega_2 \implies \omega_2 \leq \omega_1 \leq \omega_2
\]

Entropy between orders corresponds to inclusion of order varieties: given two order varieties \( \alpha, \beta \) on \( X \) and \( x \in X \), we have

\[
\alpha \subseteq \beta \quad \text{iff} \quad (\alpha)_x \sqsubseteq (\beta)_x
\]

This is independent of the choice of \( x \). Entropy is performed in the tree representation for series-parallel order varieties by changing some \( \leq \) nodes into \( \uparrow \) nodes, i.e. by weakening the information on the nodes.

2.4 Splitting

Splitting is the following problem: given series-parallel order varieties \( \alpha, \beta \) respectively on \( \{1, \ldots, n\} \) and on \( X \), and a partition \( (X_1, \ldots, X_n) \) of \( X \) (i.e. \( \bigcup X_i = X \)), find series-parallel orders \( \{\omega_1, \ldots, \omega_n\} \) such that \( \beta \sqsubseteq \alpha(\omega_1, \ldots, \omega_n) \).

This problem has been solved in [8] for the case of the binary splitting and then generalized in [9] to the case of the \( n \)-ary splitting:

**Theorem 1** The splitting problem for \( \alpha, \beta, (X_i)_{i=1, \ldots, n} \), has a solution if and only if the following condition (known as admissibility) holds:

\[
\forall i \in X_i, \ b \in X_j, \ c \in X_k / \ i \neq j \neq k \neq i \quad \beta(a, b, c) \Rightarrow \alpha(i, j, k) \\
\forall i, \ b \in X_i, \ c \in X_j, \ c' \in X_k / \ i, j, k \neq i \quad \neg(\beta(a, b, c) \land \beta(b, c, c'))
\]

When this condition holds, the set of solutions is given by \( \tau_i \sqsubseteq \omega_i \) for all \( i \), where

\[
\tau_i = \bigcup_{a \in X_i, \ j \neq i} \alpha_j|_X_i
\]

2.5 Connectives

Formulas are built from (negative) atoms \( a, b, \ldots \) and their (positive) duals \( a^\perp, b^\perp, \ldots \), and the following connectives:

<table>
<thead>
<tr>
<th>Multiplicatives</th>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \top )</td>
<td>one</td>
<td>( \bot )</td>
</tr>
<tr>
<td>( \otimes )</td>
<td>times</td>
<td>( \forall )</td>
</tr>
<tr>
<td>( \oplus )</td>
<td>next</td>
<td>( \sigma )</td>
</tr>
<tr>
<td>( ! )</td>
<td>of-course</td>
<td>( ? )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Additives</th>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \top )</td>
<td>top</td>
<td>( 0 )</td>
</tr>
<tr>
<td>( &amp; )</td>
<td>with</td>
<td>( \oplus )</td>
</tr>
<tr>
<td>( \forall )</td>
<td>forall</td>
<td>( \exists )</td>
</tr>
</tbody>
</table>

Duality is defined by usual De Morgan rules. For instance, \( (A \otimes B)^\perp = B^\perp \lor A^\perp \). The sequent system of Non Commutative logic is described in Appendix A.
3 Constraint Based Proof Construction

A generic, constraint based proof construction procedure has been proposed for Linear Logic in [3]. It is recalled here and presented in a way that facilitates its extension to the cyclic and non-commutative cases, presented in the next sections.

3.1 Outline of the Generic Approach

One of the objectives of the constraint based approach to proof construction is to be able to deal with partial information about the object being constructed. This object is therefore not a ground proof, but an “abstract proof” describing a non-empty set of ground proofs derived from a common pattern. Typically, an abstract proof consists of a tree structure, labeled by pairwise distinct variable identifiers, called its main variables, together with a set of equations linking those variables. The equations are collected during the proof construction process. They may involve side variables which do not explicitly appear as labels of the tree. The main variables range over the set of sequents, so that each solution of the equation system corresponds exactly to a ground proof, obtained by replacing in the tree each main variable by its corresponding sequent in the solution. The set of ground proofs attached to an abstract proof can therefore be identified to the set of solutions of its equation system.

In a traditional logic programming setting, one already manipulates such abstract proofs. For example, when using Horn clauses, one can work in a sequent system in which the sequents are always of the form \( \mathcal{P} \vdash a \), where \( \mathcal{P} \) is a fixed set of clauses (the program) and \( a \) is an atomic goal. The equations are therefore of the form: \( X = (\mathcal{P} \vdash a) \). The only source of partial information in that case comes from the fact that \( a \) may not be entirely known: it may contain side variables which range over first-order terms. Unification is the process that solves equation systems of that form.

In general, the complete resolution of the equation system of an abstract proofs is never actually performed. However, one wants to make sure that, at any stage of the proof construction process, the equation system remains consistent, and one could conceptually compute all the solutions on demand, even if this is never actually done. Hence the need for a resolution procedure for equation systems, similar to unification, which basically performs rewriting steps on the input system, each step preserving the set of solutions in order to achieve a system from which the solutions can be straightforwardly computed. When dealing with abstract proofs in general, however, the rewriting steps of a resolution procedure, unlike unification, may not all be deterministic: some steps are allowed to make non-deterministic choices, so that the resolution procedure builds a tree or “tableau”), instead of a sequence, of equation systems. The nodes of such a tableau are called states. A state is said to be open if it has not been expanded yet. The output of the resolution procedure is a tableau \( \mathcal{T} \) which satisfies the following requirements:

- The root state of \( \mathcal{T} \) is labeled with the input equation system.
- **Preservation**: Each expansion of a state in \( \mathcal{T} \) must preserve the set of solutions. In other words, restricted to the main variables, the set of solutions of the equation system at one non-open state of \( \mathcal{T} \) must be equal to the union of the sets of solutions of the equation systems at each of its successors.
- **Resolution**: The open states of \( \mathcal{T} \) must be labeled with equation systems in solved form, i.e., containing only equations of the form \( X = c \) where \( X \) is a main variable and \( c \) is a ground sequent, from which it is straightforward to obtain a solution.
- **Termination**: \( \mathcal{T} \) must have a finite depth, meaning that the resolution procedure performs only bounded sequences of rewriting steps to reach any of the solutions. However, \( \mathcal{T} \) may be of infinite width, meaning that some of the choices at the non deterministic steps may involve infinitely many alternatives.

In order to be a decision procedure, the resolution procedure is split into two phases, which may execute different sets of rewrite rules:

- The “Simplification” phase starts from the root state and must yield a finite tableau, the open states of which are all consistent. Given the Termination property (bounded depth), being finite means that all the choices performed during that phase involve only finite alternatives (bounded width).
- The “Generation” phase continues the expansion of the tableau after the Simplification phase, but all the states created during this phase must be consistent. Since the Simplification phase produces only consistent open states, this means that the Generation never makes choices that lead to an inconsistency (no dead-end).

These two phases are illustrated in Figure 1. Simplification defines a decision procedure since the consistency of the root state is decided by the existence of an open state at the outcome of the Simplification phase which requires only a finite number of finite choices, which can therefore be fully enumerated, e.g., by a systematic tree traversing procedure.
The treatment of equations of the form \( t_1 = t_2 \) between first order terms gives an idea of the Simplification and Generation procedures in that case.

- In the Simplification phase, we have the usual rewrite rules for unification:

  \[
  \begin{align*}
  [f(t_1, \ldots, t_n) = g(u_1, \ldots, u_m)] & : \mathcal{E} \rightarrow \text{DEADEND if } f \text{ and } g \text{ are distinct function symbols} \\
  [f(t_1, \ldots, t_n) = f(u_1, \ldots, u_m)] & : \mathcal{E} \rightarrow \{ t_1 = u_1 \} : \ldots : \{ t_n = u_n \} : \mathcal{E} \\
  [x = t] & : \mathcal{E} \rightarrow \text{DEADEND if } x \text{ occurs inside } t \\
  [x = t] & : \mathcal{E} \rightarrow (x := t)\mathcal{E} \text{ if } x \text{ does not occur inside } t
  \end{align*}
  \]

  where \( f \) and \( g \) are function symbols of respective arities \( n \) and \( m \). The equation system obtained from \( \mathcal{E} \) by substituting \( x \) by \( t \) is denoted \( (x := t)\mathcal{E} \).

- In the Generation phase, we have the following rewrite rule: a tableau with equation system \( \mathcal{E} \) containing a first-order variable \( x \) is expanded into one new state for each arbitrary choice of ground term \( t \), with the equation system \( (x := t)\mathcal{E} \).

Note that while Generation is obviously highly non-deterministic (arbitrary choice of a ground term at each step), Simplification never performs choices with multiple alternatives. The tableau in this phase has a single branch. However, strictly speaking, Simplification is not deterministic, since there may be no alternative at all (deadend or failure case). We introduce below fully non-deterministic Simplification rules with multiple alternatives.

### 3.2 The Focussing Bipolar Sequent Calculus

Proof construction in a sequent calculus is intrinsically a non-deterministic process: at each step, some new information is added to the object being built, but this can be done in several ways. It is however well known that, due to intrinsic permutability properties of sequent systems, some strategy is needed to avoid making sets of choices which lead to the same object (modulo permutation of inferences). In the case of Non Commutative Logic, just as in Linear Logic, such a strategy can be expressed by the Focussing property, together with the restriction to a certain type of formulas called bipoles, defined below. These definitions are straightforward extensions to the Non Commutative case of the corresponding definitions given for Linear Logic in [3].

**Definition 1** Monopoles and bipoles are classes of formulas inductively defined as follows:

- A **monopole** is a formula built from the negative atoms, possibly prefixed with the modality \( ? \), using the negative connectives.

  \[
  M = \bot \mid M_1 \land M_2 \mid M_1 \lor M_2 \mid T \mid M_1 \& M_2 \mid \forall x.M \mid ?a \mid a
  \]
• A bipole is a formula built from the monopoles, possibly prefixed with the modality !, and the positive atoms, using the positive connectives.

\[ B = 1 | B_1 \otimes B_2 | B_1 \circ B_2 | 0 | B_1 \oplus B_2 | \exists x B | ! M | M | a^- \]

Furthermore, it is assumed that each bipole contains at least one positive atom.

The focussing bipolar sequent calculus of Non Commutative Logic is defined as follows:

• A sequent (resp. pre-sequent) is a pair \( \Theta : \alpha (resp. \Theta : \omega) \) where \( \Theta, \alpha, \omega \) are, respectively, a set, an order variety and an order on negative atoms. The set of all the pre-sequents (resp. sequents) is denoted \( S \) (resp. \( T \)).

• An inference (resp. pre-inference) is a pair \( \Xi \) where \( \Xi \) is a finite set of sequents (premisses) and the conclusion \( \alpha \) is a sequent (resp. pre-sequent). The set of all the pre-inferences (resp. inferences) is denoted \( I \) (resp. \( F \)).

• For each bipole \( F \), we define (below) a set \( \| F \| \) of inferences. The focussing bipolar sequent calculus is defined by the inferences in the union of the sets \( \| F \| \) where \( F \) ranges over bipoles.

For each bipole \( F \), the set \( \| F \| \) contains all the inferences which capture in a single step the complete decomposition of \( F \) in the context of the conclusion of the inference. This operation can be defined in two stages which correspond to the decomposition of each of the layers of the bipole. For purpose of clarity, in the sequel, we will not consider the (first-order) quantification, although all the results can be easily adapted to this case.

**Definition 2** For any bipole \( F \), the set \( \| F \| \) is defined in four steps:

• Each connective is interpreted as an operator on sets of pre-sequents and pre-inferences

\[
\begin{align*}
\downarrow : \wp(S) &\quad \nabla : \wp(S) \times \wp(S) \mapsto \wp(S) \\
\downarrow : \wp(I) &\quad \nabla : \wp(I) \times \wp(I) \mapsto \wp(I)
\end{align*}
\]

• \( \downarrow = \{ e : e \} \)

\[
\begin{align*}
\uparrow &\downarrow \downarrow S_1 \downarrow S_2 = \{ \theta, \theta_2 : \omega_1 / \theta : \omega_1 \in S_1, \quad \theta : \omega_2 \in S_2 \}, \quad i = 1, 2 \\
\uparrow &\downarrow \nabla S_1 \nabla S_2 = \{ \theta, \theta_2 : \omega_1 < \omega_2 / \theta : \omega_1 \in S_1, \quad \theta : \omega_2 \in S_2 \}, \quad i = 1, 2
\end{align*}
\]

\( \uparrow F \) is defined by the inferences in the union of the sets \( \| F \| \) where \( F \) ranges over bipoles.

• A mapping \( M \mapsto M^- \) from monopoles into the set of pre-sequents \( \wp(S) \) is inductively defined as follows:

\[
\begin{align*}
1^- &\downarrow = 1 \\
(M_1 \circ M_2)^- &\nabla = M_1^- \nabla M_2^- \quad (\exists x F)^- &\nabla = (\exists x F^-)^- \\
(\forall \Theta : \alpha)^- &\nabla = (\forall \Theta : \alpha^-)^- \quad (\forall \Theta : \alpha)^- &\nabla = (\forall \Theta : \alpha^-)^-
\end{align*}
\]

where \( \alpha \) is a negative atom and \( M_1, M_2 \) are monopoles.

• A mapping \( B \mapsto B^+ \) from bipoles into the set of pre-inferences \( \wp(I) \) is inductively defined as follows:

\[
\begin{align*}
1^+ &\downarrow = 1 \\
(B_1 \circ B_2)^+ &\circ = B_1^+ \circ B_2^+ \\
(B_1 \circ B_2)^+ &\circ = B_1^+ \circ B_2^+ \\
(\exists M)^+ &\circ = ! (M^+)^- \\
(\forall \Theta : \alpha)^- &\nabla = (\forall \Theta : \alpha^-)^- \\
(\forall \Theta : \alpha)^- &\nabla = (\forall \Theta : \alpha^-)^-
\end{align*}
\]

where \( \alpha \) is a negative atom, \( M \) is a monopole and \( B_1, B_2 \) are bipoles.
• Finally, for any bipole $F$ we have

$$\|F\| = \{ \frac{\Theta}{\Theta^\prime}, \alpha : \Theta < \Theta^\prime \land \alpha \subseteq \Omega \}$$

Note that the conditions $\Theta \subseteq \Theta^\prime$ and $\alpha \subseteq \Omega$ in the definition of $\|F\|$ capture, respectively, the structural rules of Weakening (Contraction is implicit in the fact that $\Theta, \Theta^\prime$ are sets) and Entropy. Thus, in the focusing bipolar calculus, the structural rules never occur in the decomposition of the bipoles themselves, only at the beginning of such decompositions.

**Example 1** Consider the following bipole

$$F = p^+ \circ ((|a|^\langle k \nu e \rangle) \& \langle d' \nu (f' ? c') \rangle) \circ q^+ \circ (r^+ \circ ! (d' \nu e) \circ (f' \nu g))$$

Applying the definitions above, we have

$$M_{\Gamma}^+ = \{ e : a, b' < b \} \quad M_{\Theta}^+ = \{ \Theta_1 : \Theta_2, \Theta_3, p^+ : d \}$$

$$M_{\Theta}^- = \{ e : d \} \quad M_{\Theta}^- = \{ \Theta_2 : \Theta_3, p^+ : d \}$$

$$M_{\Theta}^- = \{ e : f < g \} \quad M_{\Theta}^- = \{ \Theta_2 : \Theta_3, f < g \}$$

where $\Theta_1, \Theta_2, \Theta_3$ range over sets, and $\Omega, \Omega_1, \Omega_2$ over orders ($\Omega_2 = \varepsilon$ because of the exponential). Now $(p^+)^+$ consists of the axioms $e : p$ and $p^+ : p'$ where one of $p', p''$ is $p$ and the other is $e$ (we make the same convention for $q$ and $r$). Hence, $F^+$ consists of the pre-inferences of the form

$$\frac{\Theta_1 : \Theta_2, \Theta_3, p^+ : d \circ ! (d' \nu e) \circ (f' \nu g)}{\Theta_1, \Theta_2, \Theta_3, p^+ : d \circ ! (d' \nu e) \circ (f' \nu g)}$$

Finally, $\|F\|$ is the set of inferences of the form

$$\frac{\Theta_1 : \Theta_2, \Theta_3, p^+ : d \circ ! (d' \nu e) \circ (f' \nu g)}{\Theta_1, \Theta_2, \Theta_3, p^+ : d \circ ! (d' \nu e) \circ (f' \nu g)}$$

where $\Theta_1, \Theta_2, \Theta_3$ are sets, $\Omega_1, \Omega_2$ are orders and $\alpha$ is an order variety (on negative atoms) such that:

$$\frac{\Theta_1 \cup \Theta_2 \cup \Theta_3 \subseteq \Theta}{\alpha \subseteq \Omega \circ \|F' \circ (f' \nu \Omega_2)\|}$$

where $p'$ (resp. $q', r'$) is $p$ (resp. $q, r$) or $e$, but in the latter case $p$ (resp. $q, r$) must be in $\Theta$.

It has been shown in [3], in the case of Linear logic, that proofs in the standard sequent calculus are essentially equivalent (modulo permutation and grouping of inferences, and renaming of formulas) to proofs in the focusing bipolar sequent calculus. This result extends straightforwardly to Non Commutative Logic (see Appendix A for some details), so that proof construction can be performed, without loss of generality, in the focusing bipolar sequent calculus as defined above. We now apply the generic constraint based approach outlined in Section 3.1 to this calculus.

### 3.3 Abstract Proofs in the Focusing Bipolar Sequent Calculus

In the focusing bipolar sequent calculus, an abstract proof is made of abstract inferences of the form

$$\frac{X_1 \ldots X_n}{X} F$$

where $X_1, \ldots, X_n$ are main variables and $F$ is a bipole. The equation system attached to such an abstract inference must express that, once the variables are replaced by their values (actual sequents), the resulting inference is an element of $\|F\|$. We now give a method to build this equation system for any given bipole $F$. We first simplify the problem by reducing it to the case of $\varepsilon$-free bipoles. Let an "alternative" of $F$ be a formula obtained from $F$ by recursively replacing each sub-formula $G_1 \equiv G_2$ by either $G_1$ or $G_2$ till no such sub-formula occurs. It is easy to show that if $F_1, \ldots, F_n$ are all the alternatives of $F$, then

• each $F_i$ is a $\varepsilon$-free bipole, and
• \( \| F \| = \bigcup_{i=1}^{n} \| F_i \| \), hence the equation system of \( F \) is the disjunction of the equation systems of \( F_i \).

For simplicity sake, we will work in the multiplicative-additive fragment, thus ignoring the exponentials, although the method presented below could be adapted to handle them. Under this restriction, the sequents (resp. pre-sequents) become simply order varieties (resp. orders) on negative atoms. Now, it is easy to characterise by a set of equations the inferences in \( \| F \| \) for a given \( \oplus \)–free bipolar \( F \). This is done in two steps.

1. We first introduce the form of the equations that we use. The equation system introduced by one bipolar is modeled as a pair \((\tau, \phi)\) such that:

   • \( \tau \) is an order;
   • \( \phi \) is a mapping which maps each \( i \in \tau \) into either a positive atom or a set of pre-sequents (orders on negative atoms). If \( \phi(i) \) is a positive atom, \( i \) is said to be terminal.

The equation system described by the pair \((\tau, \phi)\) makes use of a separate side variable for each \( i \in \tau \), a separate main variable for the conclusion and a separate main variable for each premise: there is one premise for each pair \( i, \omega \) where \( i \in \tau \) is non-terminal and \( \omega \in \phi(i) \). The equations are:

<table>
<thead>
<tr>
<th>conclusion</th>
<th>( X \subseteq \bigcap_{i \in \tau} | Y_i |<em>{\phi^{-1}} ) where ( X ) is the main variable introduced for the conclusion and ( { Y_i }</em>{i \in \tau} ) are all the side variables.</th>
</tr>
</thead>
<tbody>
<tr>
<td>non-terminal premiss</td>
<td>( X = \omega \ast Y ) where ( X ) is the main variable introduced for premise ( i, \omega ), and ( Y ) is the side variable introduced for ( i ); there is one such equation for each pair ( i, \omega ) such that ( i \in \tau ) is non-terminal and ( \omega \in \phi(i) ).</td>
</tr>
<tr>
<td>terminal premiss</td>
<td>( Y = a ) where ( a ) is the negative atom such that ( \phi(i) = a^+ ), and ( Y ) is the side variable introduced for ( i ); there is one such equation for each terminal ( i \in \tau ).</td>
</tr>
</tbody>
</table>

2. We now specify the mapping which associates to each \( \oplus \)–free bipolar \( F \) the equation system, or, equivalently, the pair \((\tau, \phi)\), describing the set of inferences \( \| F \| \). There are just five cases to consider:

   • If \( F = 1 \) then \( \tau \) is the empty order and \( \phi \) the empty mapping;
   • If \( F = F' \circ F'' \), let \( (\tau', \phi') \) and \( (\tau'', \phi'') \) be the equation systems attached to \( F' \) and \( F'' \) respectively. We assume that their support sets are renamed apart, i.e. \( |\tau'| \cap |\tau''| = \emptyset \). Then \( \tau = \tau' \circ \tau'' \) and \( \phi \) is unambiguously defined as \( \phi' \) on \( |\tau'| \) and \( \phi'' \) on \( |\tau''| \).
   • If \( F = F' \circ F'' \), the definition is the same as above except that \( \tau = \tau' \circ \tau'' \).
   • If \( F \) is a monopole, then choose an arbitrary singleton order \( \tau \) (i.e. \( |\tau| = \{ i \} \)) and let \( \phi(i) = F^- \).
   • If \( F \) is a positive atom \( a^+ \), then choose an arbitrary singleton order \( \tau \) (i.e. \( |\tau| = \{ i \} \)) and let \( \phi(i) = a^+ \).

**Example 2** Consider the bipolar

\[
F = p^+ \circ ((a \nvdash (b \nvdash c)) \circ (d \nvdash e)) \circ (r^+ \circ (d \nvdash e))
\]

Applying the definitions above, the equation system attached to \( F \) is given by the pair \((\tau, \phi)\), involving distinct side variables \( Y_1, Y_2, Y_3, Y_4, Y_5 \):

\[
\tau = Y_1 \parallel (Y_2 < Y_3 < (Y_4 \parallel Y_5))
\]

\[
\phi(Y_1) = p^+ \quad \phi(Y_2) = \{ a \parallel (b < c) , d' < e' \} \quad \phi(Y_3) = q^+ \quad \phi(Y_4) = r^+ \quad \phi(Y_5) = \{ d < e \}
\]

By elimination of the terminal side variables, the equation system describing the inferences of \( \| F \| \) becomes:

\[
\begin{align*}
X_21 & = X_2  \\
X_22 & = X_2  \\
X_3 & = X
\end{align*}
\]

**Theorem 2** Let \( F \) be a bipolar and \( \sigma_1, \ldots, \sigma_n, \sigma \) be sequents. The inference \( \sigma_1, \ldots, \sigma_n \vdash \sigma \) is in the focussing bipolar sequent calculus if and only if \( \{ X_0 = \sigma_i \}_{i=1}^{n} \) is a solution (in the main variables) of the equation system attached to \( F \).

This is shown by straightforward induction on \( \sigma \).
4 Resolution Algorithms for Non-Commutative Logic

We now come to the problem of defining a resolution algorithm for the constraint systems generated by proof construction in the focussing bipolar sequent calculus (and hence, indirectly, in non-commutative logic). Such an algorithm can be used to ensure that the constraint system at any stage in the construction is consistent. This algorithm is an instance of the generic method presented in Section 3.1. It therefore consists of a non-deterministic rewriting procedure building a tableau of equation systems, with two distinct phases of Simplification and Generation, and satisfying the properties of Preservation, Resolution and Termination. For presentation purpose, we first recall the algorithm in the commutative case viewed here as a fragment of non-commutative logic. We then show that a similar approach applies to the other important fragment: cyclic logic. Unfortunately, the method does not extend easily to the whole system, and we propose a different algorithm capable of dealing with the whole system.

4.1 The Degenerated Cases: Commutative and Cyclic Logic

4.1.1 The Commutative Case

We first consider the degenerated case where bipole uses no multiplicative connectives other than the commutative ones: \(0, \mathcal{Y}\). The equation system \(\{\tau, \phi\}\) for such a bipole has a simple form: \(\tau\) is of the form \(1 \parallel \ldots \parallel n\). The conclusion equation \(X \in \mathcal{G}(Y_1, \ldots, Y_n)\) becomes in fact a multisets equality \(X = Y_1 + \ldots + Y_n\) (indeed, remember that the inclusion concerns the structure, which is here void, not the support sets, which must be equal). By reasoning in the same way with the other equations, and by eliminating the terminal variables, we obtain exactly the type of equations identified in [3], where all the variables are multisets of atoms:

- The conclusion equation is of the form \(X = \Delta + Y_1 + \ldots + Y_n\) where \(Y_1, \ldots, Y_n\) are the non-terminal side variables \((n \geq 0)\) and \(\Delta\) is the multisets \(\alpha_1, \ldots, \alpha_m\) where \(Y_{n+1}, \ldots, Y_{n+m}\) are the terminal side variables and \(\phi(n + i) = \alpha_i^n\).

- The non-terminal premise equation for the \(i\)-th non-terminal side variable \(Y_i\) and multisets \(\Gamma \in \phi(i)\) is of the form \(X = \Gamma + Y_i\).

Thus, the overall equation system attached to an abstract proof consists of equations of the form \(X = \Gamma + Y = \Delta + Y_1 + \ldots + Y_n\) (one part comes from the equation where \(X\) is premise and one part from the equation where \(X\) is conclusion). The system can be solved in the side variables only; the solutions in the main variables are then straightforwardly derived. The resolution algorithm for such equation systems has been completely described in [3]. It deals with (oriented) equations of the following general form:

\[
\Gamma + Z_1 + \ldots + Z_m = \Delta + Y_1 + \ldots + Y_n
\]

where \(Y_1, \ldots, Y_n, Z_1, \ldots, Z_m\) are side variables ranging over multisets of atoms and \(\Gamma, \Delta\) are multisets of atoms. The Simplification phase aims at reducing the system to a system of “generators”, which are equations of this form in which \(\Delta\) is empty and \(n \geq 1\). The Generation phase then use these generators to produce all the solutions of the system. The resolution also introduces equations between atoms of the form \(a = b\). In the first-order case, they are handled by the usual unification procedure:

- Simplification procedure: main rule

\[
\begin{align*}
[\Gamma + Z_1 + \ldots + Z_m = \Delta + a + Y_1 + \ldots + Y_n] & : \mathcal{E} \\
\rightarrow & \ldots \\
\rightarrow & [\Gamma' + Z_1 + \ldots + Z_m = \Delta + Y_1 + \ldots + Y_n] : [a = b] : \mathcal{E} \quad (i) \\
\rightarrow & \ldots \\
\rightarrow & [\Gamma + Z_1 + \ldots + Z_j + \ldots + Z_m = \Delta + Y_1 + \ldots + Y_n] : [Z_j := a + Z_j'] : \mathcal{E} \quad (ii)
\end{align*}
\]

with (i) one output state for each \(b\) such that \(\Gamma = b, \Gamma'\) and (ii) one output state for each \(j = 1, \ldots, m\), introducing a fresh side variable \(Z_j\).

Thus, each element \(a\) of \(\Delta\) is sent either onto an element \(b\) of \(\Gamma\) (triggering the unification \(a = b\) with potential failure) or onto \(Z_j\) for some \(j = 1, \ldots, m\), in which case \(Z_j\) must be of the form \(a + Z_j'\). Obviously, there are finitely many alternatives for that choice.
• Simplification procedure: other rules

\[ [\Gamma + Z_1 + \ldots + Z_m = \epsilon : \mathcal{E}] \quad \rightarrow \quad \text{DEADEND} \quad \text{if } \Gamma \text{ is non empty} \]

\[ [\epsilon + Z_1 + \ldots + Z_m = \epsilon : \mathcal{E}] \quad \rightarrow \quad (Z_1 := \epsilon) \ldots (Z_m := \epsilon) \mathcal{E} \]

• Generation procedure: if \( n \geq 1 \) and \( Z_1, \ldots, Z_m \) do not occur in \( \mathcal{E} \)

\[
[[\Gamma + Z_1 + \ldots + Z_m = \epsilon + Y_1 + \ldots + Y_n : \mathcal{E}] \quad \rightarrow \quad \begin{cases}
\cdots \\
(\Gamma := \Delta_1) \ldots (Y_n := \Delta_n) \mathcal{E}
\end{cases}
\]

with (i) one output state for each arbitrary choice of multisets \( \Gamma, \ldots, \Gamma_m \) and each multiset partition \( \Delta_1, \ldots, \Delta_m \) of \( \Gamma + \Gamma_1 + \ldots + \Gamma_m \). Thus, Generation just works as Simplification, but in a reverse direction: each element of \( \Gamma + \Gamma_1 + \ldots + \Gamma_m \) is sent onto \( Y_i \) for some \( i = 1, \ldots, n \). Note that, assuming \( n \geq 1 \), there is a finite but non-null number of ways to partition any given multiset into \( n \) pieces. If \( n = 0 \) was allowed, the partition would be impossible. On the other hand, if \( m \geq 1 \), there are infinitely many alternatives for the choice of a \( m \)-uple of multisets, and a single one if \( m = 0 \).

It is shown in [3] that the Simplification phase produces a finite tableau with no inconsistent open states, that the Generation phases produces only consistent states, and that the overall resolution procedure satisfies the expected properties (Preservation, Termination, Resolution). All these properties hold only if the initial equation system is obtained from an abstract proof, ensuring a certain regularity in the shape of the system (see [3] for a precise definition of “regularity”): the resolution algorithm defined here is not a general multiset equation resolution procedure. In particular, with regularity, the Simplification procedure propagates information downwards in the (finite) abstract proof, so that the output tableau is always bounded by the size of the abstract proof.

4.1.2 The Cyclic Case

We now consider the second degenerated case where the bipoles involve no multiplicative connectives other than the non-commutative ones: \( \odot, \triangledown \). The equation system \( (\tau, \phi) \) for such a bipoles has a simple form: \( \tau \) is a total order of the form \( 1 < \ldots < n \). Reasoning as in the commutative case, we now obtain the following type of equations on lists of atoms:

\[ X = \Gamma \cdot Y = \Delta_1 \ldots \Delta_n \prec Y_1 \ldots Y_n \]

where \( \Gamma, \Delta_1, \ldots, \Delta_n \) are lists of atoms (because of the non-commutativity, it is impossible to regroup all the atoms corresponding to terminal variables into a single \( \Delta \)). Omitting the main variables, the equation above can be written \( \Gamma Y \equiv \Delta_1 Y_1 \ldots \Delta_n Y_n \), where two lists are equivalent modulo \( \equiv \) if they define the same cycle (the list concatenation operator is simply omitted in the expressions). More generally, the resolution algorithm manipulates (oriented) equations of the form

\[ \Gamma_1 Z_1 \ldots \Gamma_m Z_m \equiv \Delta_1 Y_1 \ldots \Delta_n Y_n \]

where \( Z_1, \ldots, Z_m, Y_1, \ldots, Y_n \) are side variables ranging over lists of atoms, and \( \Gamma_1, \ldots, \Gamma_m, \Delta_1, \ldots, \Delta_n \) are lists of atoms. The resolution algorithm for such equation systems is similar to that in the commutative case.

• Simplification procedure: main rule

\[
[[\Gamma_1 Z_1 \ldots \Gamma_m Z_m = \Delta_1 Y_1 \ldots \Delta_n Y_n : \mathcal{E}] \quad \rightarrow \quad \begin{cases}
\cdots \\
(\Gamma_1 := \Delta_1) \ldots (\Gamma_m := \Delta_n) \mathcal{E}
\end{cases}
\]

with (i) one output state for each \( j = 1, \ldots, m \) and each \( b \) such that \( \Gamma_j = \Gamma_j \cdot [Y_1 \ldots Y_n] \), and (ii) one output state for each \( j = 1, \ldots, m \), introducing fresh side variables \( Z_j, Z_j' \).

Thus, again, each element \( \Delta_1 \ldots \Delta_n \) is sent either onto an element \( b \) of \( \Gamma_j \) (for some \( j = 1, \ldots, m \), triggering the unification \( a = b \) with potential failure) or onto \( Z_j \) (for some \( j = 1, \ldots, m \), in which case \( Z_j \) must be of the form \( Z_j' a Z_j'' \).
• Simplification procedure: other rules

\[ [\Gamma_1 Z_1 \ldots \Gamma_m Z_m \equiv \varepsilon ; \mathcal{E}] \rightarrow \text{DEADEND if } \Gamma_1 \ldots \Gamma_m \text{ is non empty} \]

\[ [\varepsilon Z_1 \ldots \varepsilon Z_m \equiv \varepsilon ; \mathcal{E}] \rightarrow (Y_1 := \varepsilon) \ldots (Z_m := \varepsilon) \mathcal{E} \]

• Generation procedure: if \( n \geq 1 \) and \( Z_1, \ldots, Z_m \) do not occur in \( \mathcal{E} \)

\[ [\Gamma_1 Z_1 \ldots \Gamma_m Z_m \equiv \varepsilon Y_1 \ldots \varepsilon Y_n ; \mathcal{E}] \rightarrow \begin{cases} \ldots & (Y_1 := \Delta_1) \ldots (Y_n := \Delta_n) \mathcal{E} \end{cases} \]

\( \text{with } (i) \text{ one output state for each arbitrary choice of lists } \Gamma'_1, \ldots, \Gamma'_m \text{ and each partition } \text{modulo cyclicity} \Delta_1, \ldots, \Delta_n \text{ of } \Gamma_1 \Gamma'_1 \ldots \Gamma_m \Gamma'_m. \)

Following exactly the same kind of argument as in \([3]\), it can be shown that the Simplification phase produces a finite tableau with consistent open states (if any), that the Generation phases produces only consistent states, and that the overall resolution procedure satisfies the expected properties (Preservation, Termination, Resolution). Again, this assumes that the initial equation system is obtained from an abstract proof, not just any equation system on lists.

### 4.2 The General Case

In the general case, we make two essential assumptions:

- We work in affine logic. Consider indeed the following example consisting of a single node:

\[ c \parallel (a < b) \ast Y = X \parallel (a \parallel b) \ast Y. \]

After Simplification, say both occurrences of \( a \) have been matched together and so have both occurrences of \( b, c \) has not been placed in \( c \parallel (a < b) \), and only two points have been through \( X \), so no information on orders has been propagated. But if during generation we would try to place \( c \), we would get a contradiction since, clearly, it can only go into \( Y \) and the restriction of the constraint to \( a, b, c \) would be violated:

\[ c \parallel (a < b) \not\subseteq (a \parallel b) \ast c. \]

Hence Weakening is essential in the Simplification procedure to deal only with the atoms which are truly involved in the proofs, i.e. those whose dual appear in the bipoles of the proof. The others can always be always be weakened by weakening. In affine logic, the equation systems for the focussing bipolar sequent calculus are slightly altered as follows: the conclusion equations become of the form \( X \in \mathbb{S}(Y_1, \ldots, Y_n) \) where \( \mathbb{S} \) relation is defined as follows for any order varieties \( \alpha, \beta \):

\[ \alpha \subseteq \beta \text{ if and only if } D = |\beta| \subseteq |\alpha| \text{ and } \alpha|_D \subseteq \beta \]

- We take a locative view of sequent systems. This means that, like in ludics \([9]\), equations are not interpreted “modulo isomorphism” but express real equalities and inclusions between support sets. Thus, each equation becomes one equation on support sets and one equation on orders:

  - Non terminal premise equations of the form \( X = \omega \ast Y \) become:

\[
|X| = |\omega| + |Y| \\
X = \mathcal{S} \ast \omega[Y] + |\omega|Y + T
\]

(1)

(2)

For each \( u \in \omega \), called a provider place, if \( a \) is the atom at place \( u \) in \( \omega \), we add the “typing” constraint \( u : a \).

- Terminal premise equations of the form \( \omega = \alpha \) become \( Y = z \) for some place \( z \), called a consumer place, and we add the “typing” constraint \( z : a \).

- Conclusion equations of the form \( X \in \alpha(Y_1, \ldots, Y_n) \) where \( \alpha \) is a variety over \( \{1, \ldots, n\} \) become:

\[
|X| \supseteq \sum_i |Y_i| \\
X \subseteq \sum_i \mathcal{S} + \sum_{i < j} |Y_i|Y_j + \sum_{\alpha \subseteq \beta} |Y_i|Y_j + \mathcal{S}
\]

(3)

(4)
We also assume that all the provider places coming from distinct equations are pairwise distinct.

In the case of non-commutative logic, it is not possible, as in commutative and cyclic logic, to express the Resolution procedure as rewrite rules directly on the equation systems. The equation systems remain untouched, but they are augmented with derived places of information called “infons”, and the rewrite rules are expressed on multisets of those infons. The infons are of the following form:

\[
[ z \in [X] ] ; [ z \in [Y] ] ; [ z = u ] \\
[X(z_1, z_2, z_3)] ; [Y(z_1, z_2, z_3)] ; [Y(z_1, z_2)]
\]

where \( z, z_1, z_2, z_3 \) are consumer places, \( u \) is a producer place, \( X \) is a main variable (ranging over order varieties) and \( Y \) is a side variable (ranging over orders). The meaning of each infon is self-explanatory, so it is clear what a “solution” to a set of infons means. Each infon may be superscripted as \([\ldots]^*\): the meaning is the same, but this is used to control the rewriting, in particular to ensure termination.

4.3 The Simplification Procedure

Given an equation system obtained from an abstract proof in non-commutative (affine, locative) logic, the Simplification procedure starts with a single state labeled \( \text{Start} \). It then proceeds by applying the rewrite rules of Figure 2 and 3. The rules of Figure 2 propagate support set infons, in essentially the same way as in the commutative or cyclic case, i.e., downwards in the abstract proof tree. The rules of Figure 3, on the other hand, deal with order infons and propagate them upwards, but only when sufficient support set information has been propagated downwards. Of course, some rules are non-deterministic: they may have zero (deadend), one or more output states. Note that rule \( PO \) simply extends its input state, while all the other rules consume infons from their input state. The names of the rules have not been chosen arbitrarily: the first letter \( P \) or \( C \) indicates whether the rule pertains to premises or conclusion equations, respectively; the second letter \( S, O \) or \( L \) indicates whether the rule propagates support set infons, order infons or ensures linearity, respectively.

**Theorem 3** Simplification satisfies Termination, Preservation, and produces a finite tableau with no inconsistent open states.

**Proof** —

- Simplification satisfies Preservation: for example, rule \( PS \) satisfies Preservation as a direct consequence of (1). All the other cases are treated similarly: rule \( CS \) is justified by (2), rule \( PO \) by (3) and rules \( CO_i \) for \( i = 1, 2, 3 \) by (4). Rules \( PL \) and \( CL \) simply ensure linearity.

- Simplification satisfies Termination: the argument is very similar to the commutative case: the rewrite rules propagate information in a uniform direction through the abstract proof tree (towards the root for support set infons, towards the leaves for order infons), and hence all propagations must terminate. Note that the rules that do not consume any infons do not threaten Termination, since they all have a precondition that negates the presence of the infon they add, so they cannot be applied twice.

- Simplification produces a finite tableau: obvious, since all the rewrite rules involve only finite choices, and we already have shown Termination.

The main difficulty is therefore to show that the open states at the end of the Simplification procedure are consistent. This is the purpose of the rest of this section.

Let \( C \) be an open state at the end of the Simplification procedure. For any infon, we simply write \([\ldots]\) to mean \([\ldots] \in C\).

**Definition 3** Let \( Z \) be a variable ranging over \( n \)-ary cyclic relations on places (where \( n = 2, 3 \)). The \( n \)-ary relation \( Z^0 \) on places is defined by all the infons pertaining to \( Z \) inferred by the Simplification procedure:

\[
[Z] = D_Z = \{ z \in [Z]\} \\
Z(z_1, \ldots, z_n) \iff [Z(z_1, \ldots, z_n)]
\]

Let \( \mu \) be an \( n \)-ary cyclic relations (where \( n = 2, 3 \)), the \( n \)-ary relation \( \mu^0 \) is \( \mu \) restricted to the places that have effectively been involved in the Simplification procedure:

\[
[\mu^0] = \{ z \in [\mu] \land u \in [\mu] \} \\
\mu^0(z_1, \ldots, z_n) \iff [z_1 = u_1] \land \ldots \land [z_n = u_n] \land \mu(u_1, \ldots, u_n)
\]
For each premise equation \( X = \omega \times Y \),
\[
\text{PO: } \begin{cases} \text{one output state for each } u \in [\omega] \\ \rightarrow [z = 0]; [z = u]; C \end{cases} \quad \rightarrow \quad \begin{cases} \text{one output state for each } [z'] \in [Y] \\ \rightarrow [z := z']; C \end{cases}
\]

For each conclusion equation \( X \in \alpha(Y_1, \ldots, Y_n) \), and for each \( i \neq j \),
\[
\text{CS: } \begin{cases} [z \in [X]]^*; C \rightarrow [z \in [X]]^*; [z \in [X]]; C \\ \text{CL: } [z \in [Y]]; [z \in [Y]]; C \rightarrow \text{deadend}
\end{cases}
\]

Initialisation: let \( C \) be the multiset of infons of the form \([z \in [Y]]\) for each terminal premise equation \( Y = z \).
\[
\text{IS: } \begin{cases} \text{start} \rightarrow C 
\end{cases}
\]

Figure 2: Simplification rules: propagating support set information

For each premise equation \( X = \omega \times Y \),
\[
\text{CO1: } [X(z_1, z_2, z_3)]^*; C \rightarrow [Y(z_1, z_2, z_3)]; C \\
\text{if } [z_1 \in [Y]]; [z_2 \in [Y]]; [z_3 \in [Y]] \in C
\]

For each conclusion equation \( X \in \alpha(Y_1, \ldots, Y_n) \),
\[
\text{CO2: } [X(z_1, z_2, z_3)]^*; C \rightarrow [Y_1(z_1, z_2)]; C \\
\text{if } [z_1 \in [Y]]; [z_2 \in [Y]]; [z_3 \in [Y]] \in C \land i \neq j
\]

\[
\text{CO3: } [X(z_1, z_2, z_3)]^*; C \rightarrow \text{deadend} \\
\text{if } [z_1 \in [Y]]; [z_2 \in [Y]]; [z_3 \in [Y]] \in C \land i \neq j \neq k \neq i \land -\alpha(i,j,k)
\]

Figure 3: Simplification rules: propagating order information
Lemma 1 If $\mu$ is an $n$-ary cyclic relation (where $n = 2, 3$), then $\mu^\alpha$ is an $n$-ary cyclic relation which is isomorphic to $\mu|_{\alpha}$, where $\alpha = \{u \mid z = u\}$.

Proof — It is easy to show by induction that for a given consumer place $z$, there is at most one occurrence of an inf $[z = u]$ in any state. Hence $[z = u]$ defines a mapping $z \mapsto u$ from $[\mu^\alpha]$ into $\alpha \cap [\mu]$. Rule PL ensures that, in any open state at the end of Simplification, such as $\mathcal{C}_u$, this mapping is in fact a bijection (otherwise, the state would have been expanded into a deadend).

We have the following obvious consequence:

Corollary 1 If $\omega$ is an order (resp. $\alpha$ a variety) then so is $\omega^\alpha$ (resp. $\omega^\alpha$). Furthermore $\overline{\omega} = \overline{\omega^\alpha}$.

Definition 4 For each side variable $Y$, hence ranging over orders, we introduce another side variable $\overline{Y}$, ranging over order varieties, and defined by the equation $\overline{Y} = \{z \mid z \in Y\}$ where $\{z \mid z \in Y\}$ is an arbitrary new place.

Now, we extend $\mathcal{C}_u$ (modulo cyclicity) with new infs involving these new side variables:

$$\begin{align*}
\overline{[Y(z_1, z_2, z_3)]} & \to \overline{[Y(z_1, z_2, z_3)]} \\
[Y(z_1, z_2)] & \to \overline{[Y(z_1, z_2, z_3)]}
\end{align*}$$

Lemma 2 For any side variable $Y$, if the ternary relation $\overline{Y}$ is an order variety then the binary relation $Y^\alpha$ is an order, and

$$Y^\alpha = \{z \mid z \in Y\} \quad \overline{Y^\alpha} = Y^\alpha |_{\omega^\alpha}$$

Now, this holds only if $\overline{Y^\alpha}$ is an order variety. To ensure this, we add the following rewrite rules to the Simplification procedure.

$$\begin{align*}
\text{OV1:} & \quad \overline{[Y(z_1, z_2, z_3)]} \to \text{DEADEND} \\
\text{OV2:} & \quad \overline{[Y(z_1, z_2, z_3)]} \to \overline{[Y(z_1, z_2, z_3)]} \\
\text{OV3:} & \quad [z \in Y] \to \overline{[Y(z_1, z_2, z_3)]}
\end{align*}$$

Note that these extra rules do not violate the essential properties stated by Theorem 3: Termination (since the number of infs they can possibly add is limited by the total number of triples on $D_Y$), Preservation (since these rules simply express that $Y$ should be an order variety in any solution) and finiteness (since they involve only finite choices, with at most 3 alternatives). The direct consequence of these rules is:

Lemma 3 For any side variable $Y$, the ternary relation $\overline{Y}$ is an order variety.

We can now show the following central theorem:

Theorem 4 The following properties hold:

- For any conclusion equation $X \in \alpha(Y_1, \ldots, Y_n)$ we have $X^\alpha \subseteq \alpha(Y_1^\alpha, \ldots, Y_n^\alpha)$.
- For any premises equation $X = \omega * Y$ we have $\omega^\alpha * Y^\alpha \subseteq X^\alpha$.

Proof — Consider the first property: let $X \in \alpha(Y_1, \ldots, Y_n)$ be a conclusion equation.

1. Let’s first show that $[Y_i]^\alpha * [Y_j]^\alpha = \emptyset$ for $i \neq j$. Indeed, if $z \in [Y_i]^\alpha * [Y_j]^\alpha$ then we would have $[z \in Y_i] \wedge [z \in Y_j]$ and rule CL would have lead to a deadend.

2. Let $z \in [X^\alpha]$. Hence we have $[z \in X]$, which must have been obtained by rule CS from $[z \in Y_i]$ for some $i$, hence $z \in [Y_i]$. Thus, $[X] \subseteq \sum_i [Y_i]$. The converse inclusion is shown in a similar way.

3. Let $z_1, z_2, z_3 \in [X^\alpha]$. Hence we have $z_1 \in [Y_i]^\alpha \wedge z_2 \in [Y_j]^\alpha \wedge z_3 \in [Y_k]^\alpha$ for some $i, j, k$. Hence, we have $[z_1 \in Y_i] \wedge [z_2 \in Y_j] \wedge [z_3 \in Y_k]$. If $X^\alpha(z_1, z_2, z_3)$, then $[X(z_1, z_2, z_3)]$ and we have one of the following cases:

   - $i = j = k$ and, by application of rule CO1, we have $\overline{[Y_i(z_1, z_2, z_3)]}$ hence $\overline{Y_i^\alpha(z_1, z_2, z_3)}$.
   - $i = j \neq k$ and, by application of rule CO2, we have $[Y_i(z_1, z_2)]$ hence $Y_i^\alpha(z_1, z_2)$. 


\* i \neq j \neq k. Hence \( \alpha(i, j, k) \), otherwise rule CO3 would have applied, leading to a deadend, and state \( C_{\alpha} \) would not have been produced.

Thus, in all cases, we get \( \alpha(\gamma_1, \ldots, \gamma_n)(z_1, z_2, z_3) \). Thus we have shown that \( X^* \subseteq \alpha(\gamma_1, \ldots, \gamma_n) \).

We now come to the second property: let \( X = \omega * Y \) be a premises equation.

1. Let's first show that \( \omega' \cap \gamma' \neq \emptyset \). Indeed, if \( z \in \omega' \cap \gamma' \), then we would have \( [z = u] \land [z \in \gamma] \) for \( u \in \omega \), which is impossible because rule PS sends \( z \) either onto \( \omega \) or onto \( Y \), but never both.

2. Let \( z \in \omega' \cup \gamma' \). Hence, we have \( [z = u] \) for some \( u \in \omega \), or we have \( [z \in \gamma] \), which, in both cases, must have been produced by rule PS from \( [z \in \{\gamma\}] \), hence \( z \in \{X\} \). Thus, \( \omega' \cup \gamma' \subseteq \{X\} \). The converse inclusion is shown in a similar way.

3. Let \( z_1, z_2, z_3 \in \{X\} \) such that \( \omega' * \gamma' \mid (z_1, z_2, z_3) \). Hence we have one of the following cases:

   \* \( [z_1 = u_1] \land [z_2 = u_2] \land [z_3 = u_3] \) with \( u_1, u_2, u_3 \in \omega \) and \( \omega'(z_1, z_2, z_3) \). Hence, \( \omega(u_1, u_2, u_3) \) and we are in the conditions of application of rule PO.

   \* \( [z_1 = u_1] \land [z_2 = u_2] \land [z_3 = u_3] \) with \( u_1, u_2 \in \omega \) and \( \omega'(z_1, z_2, z_3) \). Hence \( \omega(u_1, u_2) \) and we are in the conditions of application of rule PO.

   \* \( [z_1 \in \gamma] \land [z_2 \in \gamma] \land [z_3 \in \gamma] \) with \( u_1 \in \omega \) and \( \gamma'(z_2, z_3) \). Hence \( \gamma(z_1, z_2, z_3) \) and we are in the conditions of application of rule PO.

   \* \( [z_1 \in \gamma] \land [z_2 \in \gamma] \land [z_3 \in \gamma] \) with \( \gamma'(z_1, z_2, z_3) \). Hence \( \gamma'(z_1, z_2, z_3) \) and we are in the conditions of application of rule PO.

Thus, in all cases, by application of rule PO, we get \( \omega * \gamma \mid (z_1, z_2, z_3) \). Thus, we have shown, \( \omega * \gamma \subseteq X^* \).

\[ \square \]

Theorem 4 ensures that \( Y = Y^* \) is a solution on the side variables at state \( C_{\alpha} \), from which it is easy to infer a solution on all the variables. Hence, we have the following corollary, which completes the proof of Theorem 3.

**Corollary 2** Simplification produces no inconsistent open states.

In fact, it is easy to see that Simplification produces the minimal solutions:

**Lemma 4** An assignment \( \sigma \) of the side variables is a solution of the set of infons at state \( C_{\alpha} \), if and only if for all side variable \( Y \) we have \( Y^* \subseteq \sigma(Y) \cap D_{\nu}. \)

**Proof** — Let \( \sigma \) be a solution of the set of infons at \( C_{\alpha} \). Hence for any side variable \( Y \), \( \sigma(Y) \) satisfies all the infons in \( C_{\alpha} \). With the support set infons, we conclude that \( D_{\nu} \subseteq \sigma(Y) \). With the order infons, we conclude that, in \( D_{\nu} \), if \( Y(z_1, z_2) \), then \( \sigma(Y(z_1, z_2)) \) and hence \( \sigma(Y)(z_1, z_2) \). The same applies for the order varieties \( Y^* \) and \( \sigma(Y) \). \[ \square \]

### 4.4 The Generation Procedure

The Generation procedure starts from each open (and hence consistent) state at the end of the Simplification procedure and proceeds by applying the rewrite rules of Figure 4 and Figure 5. We now consider a given open node \( C_{\alpha} \) at the end of the Simplification procedure. It therefore defines the sets \( D_{\nu} \) and the structures \( Z^* \) for each variable \( Z \). Using those, the Generation procedure produces additional infons of the following forms:

\[
\begin{align*}
[X]_D_{\nu} \subseteq \beta ; & \quad [X]_D_\sigma = \beta ; & \quad [Y]_D_\nu = \tau \\
[X]_D_\sigma = \beta ; & \quad [Y]_D_\nu = \tau
\end{align*}
\]

The rules of Figure 4 propagate order infons downwards in the abstract proof tree and the rules of Figure 5 propagate support set infons upwards, i.e., in both cases, in the opposite direction of the Simplification procedure (just as in the commutative or cyclic case).

**Theorem 5** Generation satisfies Resolution, Termination and Preservation, and produces no inconsistent state.

**Proof** —
* For each side variable Y, let \( \{ X_j = \omega_j \ast Y \}_{j=1}^m \) be all the premise equations involving Y.

\[
\text{PO}^+ : \quad [X_j \mid \mathcal{D}_{\omega_j} \subseteq \beta_j]_{j=1}^m \quad \rightarrow \quad \cdots \rightarrow \quad [Y \mid \mathcal{D}_y = \tau] ; [Y \mid \mathcal{D}_y = \tau] \\
\]

with one output state for each \( \tau \) such that \( Y \mid \mathcal{D}_y \subseteq \tau \) and \( \omega_j \ast \tau \subseteq \beta_j \) for all \( j = 1, \ldots, m \).

* For each conclusion equation \( X \in \alpha(Y_1, \ldots, Y_n) \).

\[
\text{CO}^+ : \quad [Y_i \mid \mathcal{D}_{\omega_i} = \tau_i]_{i=1}^n \quad \rightarrow \quad [X \mid \mathcal{D}_x \subseteq \alpha(\tau_1, \ldots, \tau_n)] \]

* Initialisation: let \( \mathcal{C} \) be the multiset of insons of the form \( [Y \mid \mathcal{D}_y = z] \) for each terminal premise equation \( Y = z \) and \( [X \mid \mathcal{D}_x \subseteq \infty] \) for each main variable \( X \) that occurs in no premise equation.

\[
\text{IO}^+ : \quad \text{START} \quad \rightarrow \quad \mathcal{C} \]

Figure 4: Generation rules: propagating order information

* For each side variable Y, let \( \{ X_j = \omega_j \ast Y \}_{j=1}^m \) be all the premise equations involving Y.

\[
\text{PS}^+ : \quad [Y = \tau] \quad \rightarrow \quad [X_j = \omega_j \ast \tau]_{j=1}^m \\
\]

* For each conclusion equation \( X \in \alpha(Y_1, \ldots, Y_n) \).

\[
\text{CS}^+ : \quad [X = \beta] ; [Y_i \mid \mathcal{D}_{\omega_i} = \tau_i]_{i=1}^n \quad \rightarrow \quad \cdots \rightarrow \quad [Y_1 = \tau_1] \cdots [Y_n = \tau_n] \]

with one output state for each \( \tau_1, \ldots, \tau_n \) such that \( \beta \in \alpha(\tau_1, \ldots, \tau_n) \) and \( \tau'_i \mid \mathcal{D}_{\omega_i} = \tau_i \) for \( i = 1, \ldots, n \).

* Initialisation: at the root node \( X \)

\[
\text{IS}^+ : \quad [X \mid \mathcal{D}_x \subseteq \beta] \quad \rightarrow \quad \cdots \rightarrow \quad [X = \beta] \\
\]

with one output state for each \( \beta \) such that \( \beta \mid \mathcal{D}_x \subseteq \beta \).

Figure 5: Generation rules: propagating support set information
• Termination and Resolution are obvious: the propagations are always performed in a uniform direction in the abstract proof tree, which is finite, so they must terminate. Furthermore, at the end, there is an infon \([Z = \ldots]\) for each variable \(Z\), so the state is in solved form.

• Preservation: let’s show it here for rule \(PO^+\). The other rules are treated in the same way.

- Let \(C\) be an output state of rule \(PO^+\) and \(\sigma\) a solution for that state. Since \(\sigma\) satisfies the infon \([Y \mid \mathcal{P}_v = \tau]\) and \(\omega^\tau \cdot \tau \subseteq \beta_j\), we have \(\omega^\tau \cdot \sigma(Y) \mid \mathcal{P}_v \subseteq \beta_j\), and since \(\sigma\) also satisfies the equation \(X_j = \omega^\tau \cdot Y\), we have \(\sigma(X_j) \mid \mathcal{P}_v \subseteq \beta_j\). Hence \(\sigma\) is a solution of the input state of the rule.

- Let \(\sigma\) be a solution for the input state of rule \(PO^+\), and let \(\tau = \sigma(Y)\mid \mathcal{P}_v\). Then by construction \(\sigma\) is a solution for the output state of the rule corresponding to that choice of \(\tau\). We just have to show that this choice is admissible, i.e., that \(Y^\tau \subseteq \tau\), which is a direct consequence of Lemma 4, and that \(\omega^\tau \cdot \tau \subseteq \beta_j\) for all \(j = 1, \ldots, m\), which is also obvious. Indeed, \(\sigma\) satisfies the equation \(X_j = \omega^\tau \cdot Y\), hence \(\sigma(X_j) \mid \mathcal{P}_v \subseteq \beta_j\). But \(\sigma\) also satisfies the infon \([X_j \mid \mathcal{P}_v \subseteq \beta_j]\), hence \(\omega^\tau \cdot \tau \subseteq \beta_j\).

Generation produces no inconsistent state: it is sufficient to show that consistency is preserved by each rewrite rule. Let’s show it here for rule \(PO^+\). The other rules are treated in the same way. Let \(\sigma\) be a solution for the input state \(C\) of rule \(PO^+\), and let \(\tau\) be such that \(Y^\tau \subseteq \tau\) and \(\omega^\tau \cdot \tau \subseteq \beta_j\) for all \(j = 1, \ldots, m\). Now, let \(\sigma'\) be defined for any side variable \(Y'\) as \(\sigma'(Y') = \sigma(Y')\) if \([Y' \mid \mathcal{P}_v = \tau'] \subseteq C\) and \(\sigma'(Y') = \tau\) for \(Y' = Y\), and \(\sigma'(Y') = Y'\) otherwise (the values for the main variable are computed accordingly, using the equations \(X' = \omega^\tau \cdot Y'\)). Now, \(\sigma'\) is precisely a solution for the output state of the rule which corresponds precisely to the choice of \(\tau\). Indeed, the inferences and equations which do not involve \([X_j \mid \mathcal{P}_v \subseteq \beta_j]\) nor \(Y\) are obviously satisfied by \(\sigma'\), either because they are satisfied by \(\sigma\) or because of Theorem 4. The inferences and equations that involve these variables are also satisfied by \(\sigma'\), because of the conditions on the choice of \(\tau\).

\[\square\]

5 Conclusion

In this paper, we have investigated the constraint based approach to proof construction in the framework of non-commutative logic. It has been shown that the method introduced for the commutative case in [3] extends quite straightforwardly to the cyclic case, but not to the whole of non-commutative logic. An alternative method, based on the explicit propagation of cyclic triples, has been introduced.

This work shows that the mechanic that operates in the propagation relies in no way on the specificity of the structures of order and order varieties. The characteristic properties of these structures only impose extra rewrite rules \((OV_i)\) for \(i = 1, 2, 3\), so that the generated information does constitute an instance of the structure (note that the choice of these extra rules is the most obvious, since it mimics exactly the definition of order varieties; it may be, however, that they are not all needed). Other structures, like simple ternary cyclic relations could also be considered.

References


Identity: $A^\perp$ is a positive atom.

$$\vdash \theta : A | A^\perp$$

Positive rules: $\omega, \tau$ are orders on positive formulas and atoms.

$$\frac{\vdash \theta : \omega | A}{\vdash \theta : [\omega > \tau] | A \otimes B} \quad \frac{\vdash \theta : \tau | B}{\vdash \theta : (\tau \parallel \omega) | A \otimes B} \quad \frac{\vdash \theta : \omega | A}{\vdash \theta : \omega s A} \quad \frac{\vdash \theta : \tau | B}{\vdash \theta : \tau s B} \quad \frac{\vdash \theta : 1}{\vdash \theta : A}$$

Negative rules

$$\frac{\vdash \theta : \omega s (A < B)}{\vdash \theta : \omega s (A \not< B)}$$

$$\frac{\vdash \theta : \omega s A}{\vdash \theta : \omega s A \& B} \quad \frac{\vdash \theta : \omega s B}{\vdash \theta : \omega s A \& B} \quad \frac{\vdash \theta : \omega s B}{\vdash \theta : \omega s \top}$$

Structural rules:

$$\frac{\vdash \theta, A : \omega s A}{\vdash \theta, A : \omega s A \& B}$$

$\vdash \theta, A : \omega s A$ absorption

$\vdash \theta : \beta$ entropy

Focussing rule: $F$ positive and $|\omega|$ contains no compound negative formulas.

$$\vdash \theta : \omega | F$$

$$\vdash \theta : \omega s F$$

Unfocussing rule: $F$ is not positive.

$$\vdash \theta : \omega s F$$

$$\vdash \theta : \omega | F$$

Table 1: NL sequent calculus

---

A Focussing sequent calculus for NL

We recall here the Focussing sequent calculus of non-commutative logic. A focussing sequent is of one of the following forms:

$$\vdash \theta : \alpha$$

where $\theta$ is a set formulas (implicitly prefixed with the why-mot exponential), and $\alpha$ is a series-parallel order variety of occurrences of formulas;

$$\vdash \theta : \omega | F$$

where $\theta$ is a set of formulas (implicitly prefixed with the why-mot exponential), $\omega$ is series-parallel order on occurrences of positive formulas and atoms, and $F$ is a single formula called the focus.

Intuitively, the meaning of the bar $|$ corresponds to the gluing of orders, so $\omega | F$ stands for $\omega s F$, but the occurrence $F$ has been syntactically distinguished. Since we can focus on any formula of an order variety, the bar marks a fixed (positive) focus and keeps track of its subformulas along the positive steps of the proof.

The inferences of the focussing sequent calculus are presented in Table 1. Note that when we build a proof, moving upwards (bottom-up), series-parallelism is preserved. The Focussing sequent calculus presented above and the Focussing Bipolar sequent calculus introduced in Section 3.2 are isomorphic. This is a straightforward extension of the corresponding result shown for linear logic in [3]. If $\mathcal{A}$ is a set of negative atoms, an $\mathcal{A}$-formula (resp. $\mathcal{A}$-sequent, $\mathcal{A}$-proof, etc.) is a formula (resp. sequent, proof, etc.) built only from the atoms of $\mathcal{A}$ and their duals.

**Theorem 6** Let $A \subseteq \mathcal{A}$ be sets of (negative) atoms and $\eta$ a bijective mapping from the $A$-formulas into $\mathcal{A}'$ whose restriction to $A$ itself is the identity. Any $\mathcal{A}'$-proof in the Focussing sequent system of non-commutative logic is isomorphic to an $\mathcal{A}$-proof in the Focussing Bipolar sequent calculus.

The isomorphism is justified by the fact that in a proof of the Focussing sequent calculus, the “critical” sections corresponding to the recursive decomposition of connectives of the same polarity can be grouped together and, by renaming using $\eta$, can be isomorphically replaced by a single inference of the Focussing Bipolar sequent calculus.
This is illustrated below. We informally define a mapping $\nu$ from positive $A$-formulas into $A'$-bipoles. Consider for instance the formula $F$ in Figure 6. The successive layers of connectives of identical polarities have been drawn on the figure. Starting from the root, each dashed line shows the border of a positive layer and each dotted line the border of a negative layer (the exponentials have a special status in that the subformula of an exponential ends a layer whatever its polarity — this corresponds to the polarity inverter that is hidden in exponentials $\Box\). Let $G$ be the formula obtained by replacing in $F$ each subformula located on the first dotted line by its image by $\eta$, which is a negative atom in $A'$. In the example, there are three such subformulas $F_1, F_2, F_3$ with topmost connectives $\Box_1, \Box_2$ and $\Box_3$ (the indices are only mentioned for reference purpose). It is easy to see that, by construction, the formula $\nu_F = \eta_1 \oplus G$ is an $A'$-biple. The set of bipoles $\nu_F$ for any positive formula $F$ is called the universal program for $\eta$. Now the bipolarisation of any $A$-proof in the Focussing sequent calculus is a proof in the Focussing Bipolar sequent calculus, involving only $A'$-atoms and bipoles from the universal program for $\eta$.

**Example 3** Here is an example of mapping from a proof in the Focussing sequent calculus of non-commutative logic into a proof in the Focussing Bipolar sequent calculus.

\[
\begin{array}{l}
\frac{\vdash d \cdot d^n}{\vdash d \cdot d^n \oplus e^n} \oplus \\
\frac{\vdash e \cdot e^n}{\vdash e \cdot d^n \oplus e^n} \oplus \\
\frac{\vdash d^n \otimes e^n < a^n \otimes (d \cdot e) \otimes a^n}{\vdash a \cdot (d \cdot e) \otimes (d^n \otimes e^n) \otimes a^n} \oplus \\
\frac{\vdash a \cdot (d \cdot e) \otimes (d^n \otimes e^n) \otimes a^n}{\vdash \eta_{1} \cdot \eta_{2} \cdot \eta_{4}} \oplus \\
\end{array}
\]

where the bipoles used in the resulting proof are given by

\[
\begin{array}{l}
B_1 = \nu(a \cdot (d \cdot e)) = \eta_{1} \oplus (a \cdot (d \cdot e)) \\
B_2 = \nu((d \cdot e) \cdot a) = \eta_{2} \oplus (a \cdot (d \cdot e)) \\
B_3 = \nu((d \cdot e) \cdot a) = \eta_{3} \oplus (a \cdot (d \cdot e)) \\
B_4 = \nu((d \cdot e) \cdot a) = \eta_{4} \oplus (a \cdot (d \cdot e)) \\
\end{array}
\]

A straightforward, practical consequence of Theorem 6 is that proof construction in non-commutative logic is equivalent to proof construction in the focussing bipolar sequent calculus.