# Proof-search in implicative linear logic as a matching problem

Philippe de Groote

LORIA UMR n$^o$ 7503 – INRIA
Campus Scientifique, B.P. 239
54506 Vandœuvre lès Nancy Cedex – France
e-mail: degroote@loria.fr

**Abstract.** We reduce the provability of fragments of multiplicative linear logic to matching problems consisting in finding a one-one-correspondence between two sets of first-order terms together with a unifier that equates the corresponding terms. According to the kind of structure to which these first-order terms belong our matching problem corresponds to provability in the implicative fragment of multiplicative linear logic, in the Lambek calculus, or in the non-associative Lambek calculus.

## 1 Introduction

Four decades ago, Lambek introduced a non-commutative logical calculus, known as **L**, intended to give a mathematical account of the structure of natural languages [13]. This calculus, which serves as a basis for modern categorial grammars [17,18,24], appears a posteriori to be the intuitionistic non-commutative fragment of Girard's multiplicative linear logic [7].

In a categorial grammar, sentence parsing amounts to automatic deduction in the underlying logical calculus. This gives a practical interest to proof-search algorithms for **L**. Nevertheless, the complexity of **L** provability is still an open problem, even in the case of its implicative fragment. It is known, however, that the calculus obtained by allowing **L** to be commutative (i.e., the intuitionistic fragment of multiplicative linear logic) is NP-complete. This result, due to Kanovitch, remains valid in the purely implicative case [9]. On the other hand, **NL** (the non-associative variant of **L** that Lambek introduced in [14]) is known to be polynomial. This has been established by Aarts and Trautwein for the implicative fragment of **NL** [1], and by ourself for the full system [6].

In this paper, we try to get some new insight into the complexity of **L**. To this end, we reduce provability in the implicative fragment of **L** to a matching problem consisting in finding a one-one-correspondence between two sets of first-order terms ranging over the free monoid, together with a unifier that equates the corresponding terms. Interestingly enough, when the terms range over the free groupoid or over the free commutative monoid, our matching problem corresponds to provability in the implicative fragments of **NL** or multiplicative linear logic, respectively. This sheds light on the role played by associativity, and commutativity.

Our reduction, which is inspired by the language models of **L** [20], is not entirely new. Indeed, in his thesis [22], Roorda shows how to associate to any formula **L** a matching problem akin to ours. With respect to this, our contribution is twofold:

- we show that a formula is provable if and only if the associated matching problem admits a solution (Roorda only proves the easy part of this statement, i.e., the necessity of the condition);
- we define a notion of PN-matching that characterises exactly the matching problems that are associated to formulas; consequently, our reduction works in both direction.

We also define a general proof-search procedure that works for the implicative fragments of **NL**, **L**, and multiplicative linear logic. This procedure, which is based on our notion of PN-matching, is specify by a non-deterministic transition system. Here, our main contribution is to show that each transition is history independent, which allows dynamic programming techniques to be used.

## 2 Intuitionistic Implicative Linear Logic

In this section, we present three variants of intuitionistic implicative linear logic: the implicative fragment of Girard's multiplicative linear logic [7], the implicative fragment of Lambek's calculus of syntactic types (also known as *the Lambek calculus*) [13], and the implicative fragment of the so-called non associative Lambek calculus [14]. These three calculi will be called **IMLL**, **IL**, and **INL**, respectively. As we will see, **IMLL** may be seen as the commutative extension of **IL** which may be seen as the associative extension of **INL**:

We start with a presentation of the weakest system. The formulas of **INL** are built up from a set of atomic formulas $\mathscr{A}$ and the connectives $\multimap$ and $\circ\!\!-$ according to the following grammar:

$$\mathscr{F} \quad ::= \quad \mathscr{A} \;\mid\; (\mathscr{F} \multimap \mathscr{F}) \;\mid\; (\mathscr{F} \circ\!\!- \mathscr{F})$$

The consequence relation of **INL** is specified by the following Gentzen-like sequent calculus. The sequents of this calculus have the form $\Gamma \vdash A$ where $\Gamma$ is a (possibly empty)[1] binary tree of formulas, i.e., a fully bracketed structure. We take for granted the notion of context, i.e., a binary tree with a hole. If $\Gamma[\,]$ is such a context, $\Gamma[\Delta]$ denotes the binary tree obtained by filling the hole in $\Gamma[\,]$ with the binary tree $\Delta$.

$$A \vdash A \quad \text{(Id)}$$

$$\frac{\Gamma \vdash A \quad \Delta[B] \vdash C}{\Delta[\,(\Gamma, (A \multimap B))\,] \vdash C} \quad (\multimap\text{-L}) \qquad\qquad \frac{(A, \Gamma) \vdash B}{\Gamma \vdash (A \multimap B)} \quad (\multimap\text{-R})$$

[1] This is a slight departure from the original (non associative) Lambek calculus that requires sequents whose antecedents are non empty.

$$\frac{\Gamma \vdash A \quad \Delta[B] \vdash C}{\Delta[\,((B \circ\!\!- A), \Gamma)\,] \vdash C} \quad (\circ\!\!-\text{-L}) \qquad \frac{(\Gamma, A) \vdash B}{\Gamma \vdash (B \circ\!\!- A)} \quad (\circ\!\!-\text{-R})$$

The above system does not include any structural rule. As a consequence **INL**, seen as logical system, is quite weak (without being trivial). For instance, the two connectives "$-\!\circ$" and "$\circ\!-$" do not satisfy the following transitivity rules:

$$(A \multimap B, B \multimap C) \vdash A \multimap C \qquad (C \circ\!\!- B, B \circ\!\!- A) \vdash C \circ\!\!- A$$

Indeed, these two rules suppose the associativity of the binary operation whose residuals are "$-\!\circ$" and "$\circ\!-$".

Now, by extending **INL** with the following structural rules, which allow for associativity, we obtain the Lambek calculus **IL**:

$$\frac{\Gamma[\,(\Delta, (\Theta, \Lambda))\,] \vdash A}{\Gamma[\,((\Delta, \Theta), \Lambda)\,] \vdash A} \quad (\text{assoc}_1) \qquad \frac{\Gamma[\,((\Delta, \Theta), \Lambda)\,] \vdash A}{\Gamma[\,(\Delta, (\Theta, \Lambda))\,] \vdash A} \quad (\text{assoc}_2)$$

To illustrate how **IL** extends **INL**, let us now derive one of the above transitivity rules:

$$\dfrac{\dfrac{\dfrac{A \vdash A \qquad \dfrac{B \vdash B \quad C \vdash C}{(B, B \multimap C) \vdash C}}{((A, A \multimap B), B \multimap C) \vdash C}}{(A, (A \multimap B, B \multimap C)) \vdash C}}{(A \multimap B, B \multimap C) \vdash A \multimap C} \quad (\text{assoc}_2)$$

In fact, the usual presentation of **IL** leaves Rules ($\text{assoc}_1$) and ($\text{assoc}_2$) implicit by defining the antecedents of the sequents to be sequences of formulas rather than binary trees.

Finally, by extending **IL** with the following exchange rule:

$$\frac{\Gamma[\,(\Delta, \Theta)\,] \vdash A}{\Gamma[\,(\Theta, \Delta)\,] \vdash A} \quad (\text{exchange})$$

one obtains the implicative fragment of Girard's multiplicative linear logic. In this case, there is no longer any need for distinguishing between two kinds of implications because the formulas $A \multimap B$ and $B \circ\!\!- A$ are provably equivalent:

$$\dfrac{\dfrac{\dfrac{A \vdash A \quad B \vdash B}{(A,\ A \multimap B) \vdash B}}{(A \multimap B,\ A) \vdash B}}{A \multimap B \vdash B \circ\!\!- A} \ (\text{exchange}) \qquad \dfrac{\dfrac{\dfrac{A \vdash A \quad B \vdash B}{(B \circ\!\!- A,\ A) \vdash B}}{(A,\ B \circ\!\!- A) \vdash B}}{B \circ\!\!- A \vdash A \multimap B} \ (\text{exchange})$$

It is well-known that **IMLL**, **IL**, and **INL** are such that any sequent $(A, \Gamma) \vdash B$ is provable if and only if $\Gamma \vdash A \multimap B$ is provable. Therefore, the provability of any sequent is equivalent to the provability of some sequent made of only one formula. In the sequel of this paper, for the sake of simplicity, we will only consider such one-formula sequents.

## 3 Intuitionistic proof-nets

In Girard's multiplicative linear logic, implication is not taken as a primitive. The formula are built upon a set of literals—i.e., atomic formulas ($A, B, C, \ldots$) or negated atomic formulas ($A^\perp, B^\perp, C^\perp, \ldots$)—by means of two connectives ($\otimes$ and $\parr$) that correspond to multiplicative conjunction and disjunction, respectively. Then, implication is defined according to de Morgan's laws. This give rise to the following translation of the implicative formulas introduced in the previous section:

$$\begin{array}{cc} [\![A]\!]^+ = A & [\![A]\!]^- = A^\perp \\ [\![\alpha \multimap \beta]\!]^+ = [\![\alpha]\!]^- \parr [\![\beta]\!]^+ & [\![\alpha \multimap \beta]\!]^- = [\![\beta]\!]^- \otimes [\![\alpha]\!]^+ \\ [\![\alpha \circ\!\!- \beta]\!]^+ = [\![\alpha]\!]^+ \parr [\![\beta]\!]^- & [\![\alpha \circ\!\!- \beta]\!]^- = [\![\beta]\!]^+ \otimes [\![\alpha]\!]^- \end{array}$$

*Example 1.* Let $F = (C \multimap (B \circ\!\!- A)) \multimap (C \multimap (B \circ\!\!- ((E \circ\!\!- E) \multimap ((D \multimap D) \multimap A))))$. Then, we have

$$[\![F]\!]^+ = ((A \otimes B^\perp) \otimes C) \parr (C^\perp \parr (B \parr ((A^\perp \otimes (D^\perp \parr D)) \otimes (E \parr E^\perp))))$$

This translation allows proof-nets to be defined for **IMLL**, **IL**, and **INL**. Proof-nets are a graph-theoretic representation of proofs. Their definition comes in two rounds. One first define a notion of proof-structure, which corresponds to a class of graphs intended to represents proofs. Then one gives a correctness criterion that allows one to distinguish the proof-structures that correspond to actual proofs from the other ones.

There exist several correctness criteria in the literature [3,4,7,8,10] (including criteria adapted to the non-commutative case [12,19,21,22]), among which the most well known are Girard's long trip condition [7], and the Danos-Regnier criterion [4]. These criteria might be used in the present intuitionistic setting because (contrarily to classical logic) multiplicative linear logic is a conservative extension of its intuitionistic fragment. Nevertheless, it is possible to define criteria that are intrinsically intuitionistic. This is the case of the criterion we give here, which is taken from [5]. This criterion has also the advantage of being easily adaptable to the non-commutative and the non-associative cases.

Proof-nets and proof-structures being simple graphs (whose vertices are decorated with literals and connectives), we use freely elementary graph-theoretic concepts that can be found in any textbook. In particular, we adopt the terminology of [2], and we will write $P = \langle V, E \rangle$ for a proof-structure (or a proof-net) $P$ whose set of vertices is $V$, and set of edges is $E$. We also take for granted the notion of parse tree of a multiplicative formula. The leaves of such a parse tree are decorated with literals, and its nodes are decorated either with the connective $\otimes$ or the connective $\parr$.

We first introduce a notion of *proof-frame*. Then we define the notions of *proof-structure* and *proof-net*.

**Definition 2.** Let $A$ be an implicative formula. The *proof-frame* of $A$ is defined to be the parse tree of the multiplicative formula $[\![A]\!]^+$. ∎

The translation $[\![\ ]\!]^+$ implicitly assigns polarities (positive or negative) to all the sub-formulas of a given implicative formula. This assignment is reflected on the proof-frames as follows:

- each leaf that is decorated with a positive literal $(A, B, C, \ldots)$ is assigned the positive polarity;
- each leaf that is decorated with a negative literal $(A^\perp, B^\perp, C^\perp, \ldots)$ is assigned the negative polarity;
- each node that is decorated with $\parr$ is assigned the positive polarity;
- each node that is decorated with $\otimes$ is assigned the negative polarity.

In a proof-frame, a subgraph made of one node together with its two daughters is called a link. The left and right daughters of a link are respectively called its left and right premises. The mother is called the conclusion of the link. Note that the two premises of any link are assigned opposite polarities by the translation $[\![\ ]\!]^+$. Consequently, one distinguishes between four sorts of links according to the polarities that are assigned to their vertices. The *npp-links* are defined to be the links whose left premise is negative, whose conclusion is positive, and whose right premise is positive. The *ppn-links*, *nnp-links*, and *pnn-links* are defined accordingly. The *npp*- and *ppn*-links are also called $\parr$-links, according to the connective that decorates their conclusions. Similarly, the *nnp*- and *pnn*-links are called $\otimes$-links.

**Definition 3.** Let $A$ be an implicative formula. A *proof-structure* of $A$ (if any) is a simple decorated graph made of:

(a) the proof-frame of $A$,
(b) a perfect matching on the leaves of this proof-frame that relates any leaf decorated with a positive literal $A$ to some leaf decorated with the negative literal $A^\perp$.

The edges defining the perfect matching on the leaves of the proof-frame are called the *axiom links* of the proof-structure. ■

In a proof-structure, the two leaves of the underlying proof-frame that are related by a given axiom link are called the conclusions of this axiom link. We also define the *principal inputs* of a proof-structure (or a proof-frame) to be the negative premises of its $\parr$-links. Similarly, We define its *principal outputs* to be the positive premises of its $\otimes$-links (this notion will be only needed in Section 4).

Let $\Sigma$ be a countably infinite set, whose elements will be called the *constants*. We write $\mathscr{T}(\Sigma)$ for the carrier set of the groupoid[2] $\langle \mathscr{T}(\Sigma), \cdot, \epsilon \rangle$ freely generated by $\Sigma$. We also write $\Sigma^*$ (respectively, $\mathbb{N}^\Sigma$) for the carrier sets of the monoid

[2] I.e, an algebraic structure with a (non necessarily associative) binary operation "$\cdot$" that admits an identity element $\epsilon$.

$\langle \Sigma^*, \cdot, \epsilon \rangle$ (respectively, the commutative monoid[3] $\langle \mathbb{N}^\Sigma, \cdot, \epsilon \rangle$) freely generated by $\Sigma$.

**Definition 4.** Let $A$ be an implicative formula, An **INL** (respectively, **IL**, **IMLL**) proof-net of $A$ (if any) is a proof-structure of $A$, $P = \langle V, E \rangle$, together with an application $\rho : V \to \mathscr{T}(\Sigma)$ (respectively, $\rho : V \to \Sigma^*$, $\rho : V \to \mathbb{N}^\Sigma$) such that:

(a) the value assigned by $\rho$ to the root of the underlying proof-frame is $\epsilon$;
(b) the values assigned by $\rho$ to the principal inputs of $P$ are constants that are pairwise different;
(c) the values assigned by $\rho$ to the two conclusions of an axiom-link are equal;
(d) the values assigned by $\rho$ obey the constraints given in Figure 1, i.e.:
  (d1) the value assigned to the positive premise of a *npp*-link must be equal to the product of the value assigned to its conclusion with the value assigned to its negative premise,
  (d2) the value assigned to the positive premise of a *ppn*-link must be equal to the product of the value assigned to its negative premise with the value assigned to its conclusion;
  (d3) the value assigned to the negative premise of a *nnp*-link must be equal to the product of the value assigned to its conclusion with the value assigned to its positive premise;
  (d4) the value assigned to the negative premise of a *pnn*-link must be equal to the product of the value assigned to its positive premise with the value assigned to its conclusion. ■

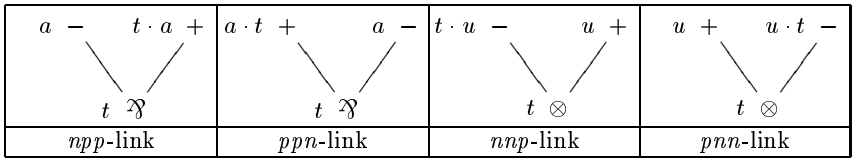



**Fig. 1.** Constraints on the links of a proof-net

In [5], the notion of *dynamic graph underlying a proof-net* is introduced. Using this notion, it is easy to prove that, for any given proof-net, the valuation $\rho$ is unique up to the renaming of the atoms assigned to the principal inputs.

*Example 5.* Figure 2 gives a proof-net for the formula of Example 1.

[3] Remark that the set of functions from a set $\Sigma$ to $\mathbb{N}$, together with the pointwise addition, corresponds indeed to the commutative monoid freely generated by $\Sigma$. Therefore, in this case, it woud be more natural to write "+" for the binary operation of the structure. Nevertheless, for the sake of uniformity, we will stick to the product notation.

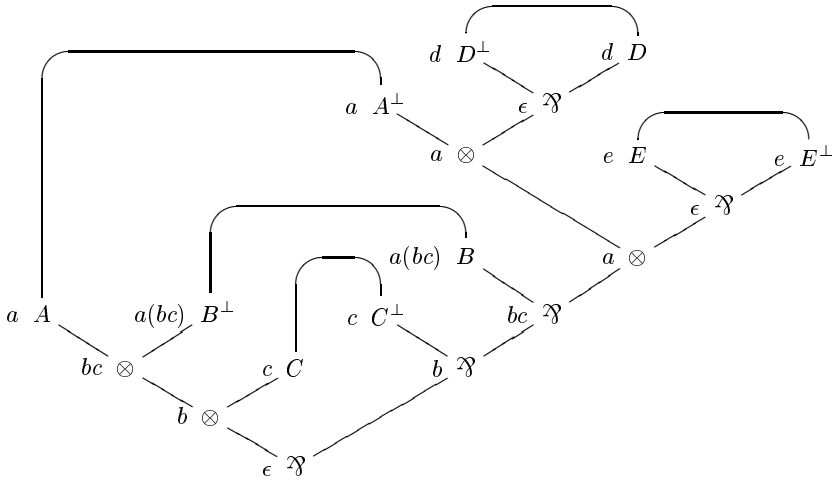



**Fig. 2.** A proof-net

In his thesis [22], Roorda noted that is possible to assign labels that obey the contraints of Definition 4 to the vertices of any correct proof-structure (i.e., a proof-structure that corresponds to some sequent derivation). On the other hand, he did not prove that the existence of such an assignement is sufficient to ensure correctness. He stated it as an open problem. We solve the question in [5] where, indeed, we proved that the condition is sufficient.[4] Consequently, we have the following proposition.

**Proposition 6.** *Let $A$ be an implicative formula. $A$ is **INL**(respectively, **IL**, **IMLL**) provable if and only if there exists an **INL**(respectively, **IL**, **IMLL**) proof-net for it.* □

## 4 Proof-search as a matching problem

Definition 4 suggests almost immediately a proof-search procedure, which may be roughly described as follows:

- given a formula, assign to the vertices of its proof-frame values that obey the constraints of Figure 1;
- try to find a set of axiom links such that the values assigned to the conclusions of any axiom link are equal.

Now, the problem in trying to assign values to a proof-frame is that the constants assigned to its principal inputs are not sufficient to determine the values assigned to its other vertices. The way out is to assign variables to some of the vertices, and then to search for a set of axiom links such that the terms assigned to

[4] In fact, Roorda conjectured that the condition was not sufficient.

the conclusions of any axiom link are unifiable. To make this idea precise, we introduce the notion of *valuated proof-frame*.

Let $\mathscr{X}$ be a countably infinite set disjoint from $\Sigma$, whose elements will be called the *variables*. We write $\mathscr{T}(\Sigma, \mathscr{X})$ for the set of terms generated by $\Sigma$ and $\mathscr{X}$ (including the identity element $\epsilon$). We have $\mathscr{T}(\Sigma) \subset \mathscr{T}(\Sigma, \mathscr{X})$ and, in this setting, the elements of $\mathscr{T}(\Sigma)$ are called the ground terms. When $t$ is a term, we write $\mathrm{var}(t)$ (respectively, $\mathrm{cst}(t)$) to denote the set of variables (respectively, constants) occurring in $t$. We extend these notations to sets of terms in the obvious way.

**Definition 7.** Let $A$ be an implicative formula. A *valuated proof-frame* of $A$ consists of the proof-frame of $A$, $P = \langle V, E\rangle$, together with an application $\rho : V \to \mathscr{T}(\Sigma, \mathscr{X})$ such that:

(a) the value assigned by $\rho$ to the root of P is $\epsilon$;
(b) the values assigned by $\rho$ to the principal inputs of $P$ are elements of $\Sigma$ that are pairwise different;
(c) the values assigned by $\rho$ to the principal outputs of $P$ are elements of $\mathscr{X}$ that are pairwise different;
(d) the values assigned by $\rho$ obey the constraints given in Figure 1. ∎

One easily shows that any formula admits a valuated proof-frame, which is unique up to a renaming of the constants assigned the principal inputs and the variables assigned to the principal outputs. Consequently, we will speak of *the* valuated proof-frame of a formula.

A substitution skeleton $\sigma$ is defined to be a partial function $\sigma : \mathscr{X} \to \mathscr{T}(\Sigma)$ whose domain is finite. Such a substitution skeleton induces a unique function $\hat{\sigma} : \mathscr{T}(\Sigma, \mathscr{X}) \to \mathscr{T}(\Sigma, \mathscr{X})$ such that:

(a) $\hat{\sigma}(\epsilon) = \epsilon$,
(b) $\hat{\sigma}(x) = \sigma(x)$, for $x \in \mathrm{dom}(\sigma)$,
(c) $\hat{\sigma}(a) = a$, for $a \in (\Sigma \cup \mathscr{X}) \setminus \mathrm{dom}(\sigma)$,
(d) $\hat{\sigma}(\alpha \cdot \beta) = \hat{\sigma}(\alpha) \cdot \hat{\sigma}(\beta)$.

A function such as $\hat{\sigma}$ is called a substitution, and we write $\mathrm{Subst}(\mathscr{X}, \Sigma)$ for the set of substitutions. By a slight abuse of language, we will speak of the domain of a substitution to mean the domain of its skeleton. If $\sigma$ and $\tau$ are two substitutions whose domains are disjoint, we have that $\sigma \circ \tau = \tau \circ \sigma$, and the skeleton of the substitution is the union of the two skeletons. In such a case, again by abuse of language, we will write $\sigma \cup \tau$ for $\sigma \circ \tau$.

The next lemma is almost immediate.

**Lemma 8.** *Let $A$ be an implicative formula, and let $P = \langle\langle V, E\rangle, \rho\rangle$ be its valuated proof-frame. Then, $A$ is **INL**(respectively, **IL**, **IMLL**) provable if and only if there exists a one-one correspondence $R$ between the positive leaves and the negative leaves of $P$, together with a substitution $\sigma \in \mathrm{Subst}(\mathscr{X}, \Sigma)$ such that for any positive leave $p$ and any negative leave $n$, $p\,R\,n$ implies that:*

(a) *if $p$ is decorated with $A$ then $n$ is decorated with $A^\perp$;*
(b) $\sigma(\rho(p)) = \sigma(\rho(n))$ *(respectively, modulo associativity, modulo associativity and commutativity).*

*Proof.* Imagine there exist such a correspondence $R$ and such a substitution $\sigma$. It is easy to check that $\langle\langle V, E \cup R\rangle, \sigma \circ \rho\rangle$ is a proof-net. Consequently, by Proposition 6, $A$ is provable.

Conversely, suppose that $A$ is provable, and consequently, that there exists a proof-net $\langle\langle V, E'\rangle, \rho'\rangle$. Take $R = E' \setminus E$ (the axiom links of the proof-nets). We have that $p\,R\,n$ implies $\rho'(p)) = \rho'(n)$. It is then easy to show, by induction on the proof-frame of $A$, that there exists a substitution such that $\rho' = \sigma \circ \rho$. □

From the above lemma, we have that to any implicative formula $A$ correspond two sets of terms $P$ and $N$ such that $A$ is provable if and only if there exists a one-one-correspondence between $P$ and $N$ together with a substitution that unifies the corresponding terms. In general, the converse is not true: one cannot associate an implicative formula to any pair of sets of terms. The main goal of this section is to characterize the pairs of sets $(P, N)$ which correspond to implicative formulas.

We define the sets of positive ($\mathscr{P}$) and negative ($\mathscr{N}$) terms as follows:

$$\begin{aligned}\mathscr{P} &::= \mathscr{X} \mid (\mathscr{P} \cdot \Sigma) \mid (\Sigma \cdot \mathscr{P})\\ \mathscr{N} &::= \Sigma \mid (\mathscr{N} \cdot \mathscr{X}) \mid (\mathscr{X} \cdot \mathscr{N})\end{aligned}$$

The unique variable occurring in a positive term is called the head of the term. Similarly, the head of a negative term is the unique constant occurring in it. We also define the set of positive ground terms ($\mathscr{G}$). These are positive terms whose head as been instantiated by a constant:

$$\mathscr{G} ::= \Sigma \mid (\mathscr{G} \cdot \Sigma) \mid (\Sigma \cdot \mathscr{G})$$

We define the accessibility relation $\prec$ on $\mathscr{G} \cup \mathscr{P} \cup \mathscr{N}$ as follows. Let $t, u \in \mathscr{G} \cup \mathscr{P} \cup \mathscr{N}$. Then, $t \prec u$ if and only if:

- either $t \in \mathscr{G} \cup \mathscr{P}$, $u \in \mathscr{N}$, and the head of $u$ occurs in $t$;
- or $t \in \mathscr{N}$, $u \in \mathscr{P}$, and the head of $u$ occurs in $t$.

We now define the central notion of this paper

**Definition 9.** A PN-matching problem consists of two finite sets of terms $P$ and $N$ such that:

(a) $P$ contains one positive ground term, called the root of the problem, and all its other elements are positive terms;
(b) all the elements of $N$ are negative terms;
(c) all the heads of the positive (respectively, negative) terms are different;
(d) the head of each positive (respectively, negative) term occurs in exactly one negative (respectively, positive or positive ground) term;

(e) each constant (respectively, variable) that occurs in a positive or positive ground (respectively, negative) term is the head of a negative (respectively, positive) term;
(f) $(\forall t \in P \cup N)\ r \prec^* t$, where $r$ is the root of the problem, and $\prec^*$ is the transitive reflexive closure of the accessibility relation.

Such a PN-matching problem admits a free-solution (respectively, A-solution, AC-solution) if and only if there exists a one-one-correspondence $R$ between $P$ and $N$ together with a substitution $\sigma \in \mathrm{Subst}(\mathscr{X}, \Sigma)$ such that $(\forall p \in P)(\forall n \in N)\ p\,R\,n$ implies $\sigma(p) = \sigma(n)$ (respectively, modulo associativity, modulo associativity and commutativity). ■

As we will see, the above notion of PN-matching corresponds to provability of one-literal formulas. It is not difficult to get rid of this restriction. It suffices to add to the problem $(P, N)$ a set of constraints $C \in P \times N$ such that

$$(\forall p_1, p_2 \in P)(\forall n_1, n_2 \in N)\,(p_1, n_1), (p_1, n_2), (p_2, n_2) \in C \Rightarrow (p_2, n_1) \in C,$$

and require that any solution $(R, \sigma)$ is such that $R \subset C$. Nevertheless, we prefer not to consider such a set of constraints $C$ in order to keep the notion of PN-matching as simple as possible. One of our goal is to gain some insight into the complexity of the Lambek calculus. With respect to this aim, there is no harm in considering only one-literal formulas. Indeed, it is a direct consequence of [15] that one-literal multiplicative formulas are not easier to prove than many-literal multiplicative formulas.

We will speak of free PN-matching, associative PN-matching, or associative commutative PN-matching according to the kind of solutions we consider (free-solution, A-solution, or AC-solution, respectively). On the other hand, when stating properties that are common to the three kinds of problems, we will simply say PN-matching.

It is not difficult to prove that the definitional properties of a PN-matching problem $(P, N)$ imply that the accessibility relation on $P \cup N$ is a tree. This property will be useful in the sequel.

Clearly, a necessary condition for a PN-matching problem to admit a solution is that $P$ and $N$ have the same cardinality. We will come back to this in Section 5.

We say that a substitution $\sigma$ is relative to a set of terms $T$ if and only if $\mathrm{dom}(\sigma) \subset \mathrm{var}(T)$. It is easy to show that, whenever a PN-matching problem $(P, N)$ admits a solution $(R, \sigma)$, it admits a solution $(R, \sigma')$ where $\sigma'$ is relative to $P$. In the sequel of this paper, we will only consider such solutions.

We end this section by proving that free, associative, and associative commutative PN-matching are equivalent to provability in **INL**, **IL**, and **IMLL**.

**Proposition 10.** *For any one-literal implicative formula $A$, there exists a* PN*-matching problem $(P, N)$ such that $A$ is **INL** (respectively, **IL**, **IMLL**) provable if and only if $(P, N)$ admits a free solution (respectively,* A*-solution,* AC*-solution).*

*Proof.* Take $P$ to be the set of terms assigned to the positive leaves of the valuated proof-frame of $A$. Similarly, take $N$ to be the set of terms assigned to its negative leaves. One may easily show, by induction on the proof-frame of $A$, that $(P, N)$ is a PN-matching problem. Then, by Lemma 8, this problem admits a solution if and only if $A$ is provable. □

As a corollary of this proposition, we have that associative commutative PN-matching is NP-complete since provability in **IMLL** is known to be NP-complete [9].

To show the converse of proposition 10, we associate a valuated partial parse tree $\mathcal{T}(t)$ to each term $t \in \mathscr{P}$ as follows:

(a) $\mathcal{T}(X)$ consists of a simple positive node $A$, which is assigned $X$;
(b) $\mathcal{T}(a \cdot t)$ is obtained as follows: replace in $\mathcal{T}(t)$ the positive leave which is assigned $t$ by a *ppn*-link whose positive premise $A$ is assigned $a \cdot t$ and whose negative premise $A^{\perp}$ is assigned $a$;
(c) $\mathcal{T}(t \cdot a)$ is obtained as follows: replace in $\mathcal{T}(t)$ the positive leave which is assigned $t$ by a *npp*-link whose negative premise $A^{\perp}$ is assigned $a$ and whose positive premise $A$ is assigned $t \cdot a$.

Similarly, one defines $\mathcal{T}(t)$, for $t \in \mathscr{N}$:

(a) $\mathcal{T}(a)$ consists of a simple negative node $A^{\perp}$, which is assigned $a$;
(b) $\mathcal{T}(X \cdot t)$ is obtained as follows: replace in $\mathcal{T}(t)$ the negative leave which is assigned $t$ by a *pnn*-link whose positive premise $A$ is assigned $X$ and whose negative premise $A^{\perp}$ is assigned $X \cdot t$;
(c) $\mathcal{T}(t \cdot X)$ is obtained as follows: replace in $\mathcal{T}(t)$ the negative leave which is assigned $t$ by a *nnp*-link whose negative premise $A^{\perp}$ is assigned $t \cdot X$ and whose positive premise $A$ is assigned $X$.

Finally, one defines $\mathcal{T}(t)$, for $t \in \mathscr{G}$:

(a) $\mathcal{T}(a)$ consists of a *npp*-link whose both premises are assigned $a$ and whose conclusion is assigned $\epsilon$;
(b) $\mathcal{T}(a \cdot t)$, where $t$ is not atomic, is obtained as follows: replace in $\mathcal{T}(t)$ the positive leave which is assigned $t$ by a *ppn*-link whose positive premise $A$ is assigned $a \cdot t$ and whose negative premise $A^{\perp}$ is assigned $a$;
(c) $\mathcal{T}(t \cdot a)$ is obtained as follows: replace in $\mathcal{T}(t)$ the positive leave which is assigned $t$ by a *npp*-link whose negative premise $A^{\perp}$ is assigned $a$ and whose positive premise $A$ is assigned $t \cdot a$.

**Proposition 11.** *For any* PN*-matching problem* $(P, N)$*, there exists a one-literal implicative formula* $A$ *such that* $A$ *is* ***INL****(respectively,* ***IL****,* ***IMLL****) provable if and only if* $(P, N)$ *admits a free solution (respectively,* A*-solution,* AC*-solution).*

*Proof.* Let $P$ and $N$ be two sets of terms that satisfiy Conditions (b), (c), (d), and (f) of Definition 9—but that does not necessarily satisfy Condition (e). We construct a valuated proof-frame $F$ by induction on the accessibility relation. If

$N$ is empty, and consequently, $P = \{r\}$ where $r$ is the root of the problem, we take $F = \mathcal{T}(r)$. Otherwise, let $t \in P \cup N$ be a term that is maximal with respect of $\prec$. Let $F'$ be the valuated proof-frame associated, by induction hypothesis, to the problem obtained by removing $t$ from $(P, N)$. $F$ is then constructed by grafting $\mathcal{T}(t)$ in place of the unique leave of $F'$ that is assigned the head of $t$ and that has the same polarity as $t$. It is not difficult to check that $F$ is indeed the valuated proof-frame of a one-literal formula $A$, and that the positive and negative leaves of $F$ are respectively assigned the elements of $P$ and $N$. Hence, the proof of the proposition follows by Lemma 8. □

As a corollary of this proposition, we have that free PN-matching is polynomial since provability in **INL** is known to be polynomial [1]. Finally, as a corollary of both Proposition 10 and 11, we have that provability of one-literal formulas in **IL** is NP-complete or polynomial if and only if associative PN-matching is NP-complete or polynomial, respectively. Consequently, the complexity problem of the Lambek calculus may be studied through our notion of PN-matching

## 5 A pn-matching algorithm

In this section, we give a general PN-matching algorithm. We first specify it by means of a non deterministic transition system. Then we explain how this algorithm may be implemented in a more efficient way.

In what follows, we assume that the terms defining a PN-matching problem are assigned different integers, and if $t$ is such an indexed term $\#t$ denotes the integer assigned to $t$. We also assume that a substitution applied on an indexed term does not affect the integer, i.e., $\#\sigma(t) = \#t$. This is only needed to keep a trace of the original terms when applying a substitution and to allow correspondences between terms to be represented as relations on integers.

Let $P$ and $N$ be two sets of indexed terms, $R \subset \mathbb{N} \times \mathbb{N}$, and $\sigma \in \mathrm{Subst}(\mathscr{X}, \Sigma)$. Consider the following transition:

$$\langle P, N, R, \sigma \rangle \longrightarrow \langle \tau(P \setminus \{t\}), \tau(N \setminus \{u\}), R \cup \{(\#t, \#u)\}, \tau \circ \sigma \rangle \qquad (1)$$

where $t \in P$ is a positive ground term, $u \in N$, and $\tau$ is a substitution whose domain is $\mathrm{var}(u)$ and such that $t = \tau(u)$.

We will prove that a PN-matching problem $(P, N)$ admits a solution $(R, \sigma)$ if and only if there exists a sequence of transitions such that:

$$\langle P, N, \varnothing, id \rangle \longrightarrow^* \langle \varnothing, \varnothing, R, \sigma \rangle \qquad (2)$$

Clearly there cannot be infinite sequences such as (2). Moreover, the branching due to the non-determinism of Transition (1) is finite. Consequently, Transition (1) specifies indeed a non deterministic algorithm. Proving the correctness of this algorithm (i.e., the if-part of the above statement) is straightforward.

**Proposition 12.** *Let $(P, N)$ be a PN-matching problem such that*

$$\langle P, N, \varnothing, id \rangle \longrightarrow^* \langle \varnothing, \varnothing, R, \sigma \rangle$$

*Then* $(R, \sigma)$ *is a solution to* $(P, N)$.

*Proof.* A straightforward induction on the sequence of transitions. □

In order to prove the completeness of the algorithm, we first establish a lemma.

**Lemma 13.** *Let* $(P_1, N_1)$ *and* $(P_2, N_2)$ *be two* PN*-matching problems such that* $\mathrm{var}(P_1) \cap \mathrm{var}(P_2) = \varnothing$. *If there exist sequences of transitions such that*

$$\langle P_1, N_1, \varnothing, id\rangle \longrightarrow^* \langle \varnothing, \varnothing, R_1, \sigma_1\rangle \quad \textit{and} \quad \langle P_2, N_2, \varnothing, id\rangle \longrightarrow^* \langle \varnothing, \varnothing, R_2, \sigma_2\rangle$$

*then there exist a sequence of transition such that*

$$\langle P_1 \cup P_2, N_1 \cup N_2, R_0, \sigma_0\rangle \longrightarrow^* \langle \varnothing, \varnothing, R_2 \cup R_1 \cup R_0, \sigma_2 \cup \sigma_1 \cup \sigma_0\rangle$$

*where* $R_0$ *is any relation, and* $\sigma_0$ *is a substitution whose domain is disjoint from* $\mathrm{var}(P_1)$ *and* $\mathrm{var}(P_2)$.

*Proof.* Since $\mathrm{var}(P_1) \cap \mathrm{var}(P_2) = \varnothing$, we have that $\sigma_1(P_2) = P_2$ and $\sigma_1(N_2) = N_2$. It is then straightforward to prove that:

$$\begin{aligned}\langle P_1 \cup P_2, N_1 \cup N_2, R_0, \sigma_0\rangle &\longrightarrow^* \langle P_2, N_2, R_1 \cup R_0, \sigma_1 \circ \sigma_0\rangle \\ &\longrightarrow^* \langle \varnothing, \varnothing, R_2 \cup R_1 \cup R_0, \sigma_2 \circ \sigma_1 \circ \sigma_0\rangle\end{aligned}$$

Moreover, we have that the domain of $\sigma_0$, $\sigma_1$, and $\sigma_2$ are pairwise disjoint. Hence, $\sigma_2 \circ \sigma_1 \circ \sigma_0 = \sigma_2 \cup \sigma_1 \cup \sigma_0$. □

We now prove the completeness of the algorithm.

**Proposition 14.** *Let* $(P, N)$ *be a* PN*-matching problem that admits a solution* $(R, \sigma)$. *Then there exists a sequence of transitions such that:*

$$\langle P, N, \varnothing, id\rangle \longrightarrow^* \langle \varnothing, \varnothing, R, \sigma\rangle$$

*Proof.* Let $r \in P$ be the root of the problem, and let $u \in N$ be such that $(\#r, \#u) \in R$. Then, let $\sigma_u$ be the substitution $\sigma$ restricted to $\mathrm{var}(u)$, and let $(t_i)_{i\in n}$ be the positive terms such that $u \prec t_i$. Define the following sets, relations, and substitutions:

$$\begin{aligned}P_i &= \{t \in P \mid \sigma_u(t_i) \prec^* \sigma_u(t)\} \\ N_i &= \{t \in N \mid \sigma_u(t_i) \prec^* \sigma_u(t)\} \\ R_i &= R \cap (\#P_i \times \#N_i) \\ \sigma_i &\text{ is the substitution } \sigma \text{ restricted to } \mathrm{var}(P_i)\end{aligned}$$

It is easy to show, from the definitional properties of a PN-matching problem that:

- $(\forall i, j \in n)$ $i \neq j$ implies $P_i \cap P_j = \varnothing$ and $N_i \cap N_j = \varnothing$,

– $\bigcup_{i\in n} P_i = P \setminus \{r\}$ and $\bigcup_{i\in n} N_i = N \setminus \{u\}$,
– $\bigcup_{i\in n} R_i = R \setminus \{(\#r, \#u)\}$ and $(\bigcup_{i\in n} \sigma_i) \cup \sigma_u = \sigma$,
– $((\sigma_u(P_i), \sigma_u(N_i)))_{i\in n}$ is a family of PN-matching problems, with $(t_i)_{i\in n}$ as roots, that admits the family of solutions $((R_i, \sigma_i))_{i\in n}$.

Then, by induction hypothesis, there exist sequences of transitions such that:

$$\langle \sigma_u(P_i), \sigma_u(N_i), \varnothing, id\rangle \longrightarrow^* \langle \varnothing, \varnothing, R_i, \sigma_i\rangle,$$

and, by iterating Lemma 13,

$$\langle \bigcup_{i\in n} \sigma_u(P_i), \bigcup_{i\in n} \sigma_u(N_i), \{(\#r, \#u)\}, \sigma_u\rangle \longrightarrow^* \langle \varnothing, \varnothing, R, \sigma\rangle,$$

which allows us to conclude since

$$\langle P, N, \varnothing, id\rangle \longrightarrow \langle \bigcup_{i\in n} \sigma_u(P_i), \bigcup_{i\in n} \sigma_u(N_i), \{(\#r, \#u)\}, \sigma_u\rangle$$

□

There are different sources of non-determinism in our PN-matching algorithm:

(a) the choice of the positive ground term $t$ according to which the transition is done,
(b) the choice of the negative term $u$ to be matched with $t$,
(c) the choice of the substitution $\tau$ such that $t = \tau(u)$.

One cannot avoid (b) and (c). On the other hand, the non-determinism due to (a) may be circumvented as we will explain by transforming our algorithm.

We first show that there is no need for updating $N$ in Transition (1). Consider the following sequence of transitions:

$$\begin{aligned}\langle P, N, \varnothing, id\rangle &\longrightarrow^* \langle P', N', R, \sigma\rangle \\ &\longrightarrow \langle \tau(P' \setminus \{t\}), \tau(N' \setminus \{u\}), R \cup \{(\#t, \#u)\}, \tau \circ \sigma\rangle\end{aligned}$$

where $(P, N)$ is a PN-matching problem. It is a direct consequence of Properties (c), (d) and (e) of Definition 9 that the substitution $\tau$ does not affect $N' \setminus \{u\}$, i.e., $\tau(N' \setminus \{u\}) = N' \setminus \{u\}$. Moreover, because of these same properties, there cannot be any positive ground term $t' \in \tau(P' \setminus \{t\})$ such that $t' = \tau'(u)$ for some substitution $\tau'$. Hence, updating $N$ is only needed in order to ensure that $P$ and $N$ have the same number of elements. But this may be checked once and for all before starting any sequence of transition. Therefore, one may assume that $N$ is an invariant datum that is global to all the possible sequences of transitions.

Now consider the set $P' \setminus \{t\}$ that appears in the above transition. This set may be partitioned into two set $P_1$ and $P_2$ as follows:

$$P_1 = \{t \in P' \mid u \prec t\} \quad \text{and} \quad P_2 = P' \setminus (P_1 \cup \{t\})$$

Again by the definitional properties of a PN-matching problem, one may prove that all the terms in $\tau(P_1)$ are positive ground terms and that $\tau(P_2) = P_2$. Moreover, using Property (f) of Definition 9, one proves that, whenever $\tau(P' \setminus \{t\})$ does not contain any positive ground term, we have $P' \setminus \{t\} = \varnothing$.

These observations lead us to the definition of a new transition:

$$\langle G, R, \sigma \rangle \xrightarrow{P,N} \langle (G \setminus \{t\}) \cup \tau(Q), R \cup \{(\#t, \#u)\}, \tau \circ \sigma \rangle \tag{3}$$

where:

- $P$, $G$, and $N$ are sets of positive, positive ground, and negative terms respectively;
- $R \subset \mathbb{N} \times \mathbb{N}$ and $\sigma \in \mathrm{Subst}(\mathscr{X}, \Sigma)$;
- $t \in G$, $u \in N$, and $\tau$ is a substitution whose domain is $\mathrm{var}(u)$ and such that $t = \tau(u)$;
- $Q = \{t \in P \mid u \prec t\}$.

It follows from the above discussion that a PN-matching problem $(P, N)$ with root $r$ admits a solution $(R, \sigma)$ if and only if there exists a sequence of transitions such that:

$$\langle \{r\}, \varnothing, id \rangle \xrightarrow{P,N}{}^{*} \langle \varnothing, R, \sigma \rangle \tag{4}$$

provided that $P$ and $N$ have the same number of elements.

Finally, let $G$, $R$, and $\sigma$ be such that

$$\langle \{r\}, \varnothing, id \rangle \xrightarrow{P,N}{}^{*} \langle G, R, \sigma \rangle \tag{5}$$

where $(P, N)$ is a PN-matching problem whose root is $r$. Assume that there exists two different transitions:

$$\langle G, R, \sigma \rangle \xrightarrow{P,N} \langle (G \setminus \{t_1\}) \cup \tau_1(Q_1), R \cup \{(\#t_1, \#u_1)\}, \tau_1 \circ \sigma \rangle$$
$$\langle G, R, \sigma \rangle \xrightarrow{P,N} \langle (G \setminus \{t_2\}) \cup \tau_2(Q_2), R \cup \{(\#t_2, \#u_2)\}, \tau_2 \circ \sigma \rangle$$

It is easy to prove, by induction on the length of Sequence (5) that $\mathrm{cst}(t_1) \cap \mathrm{cst}(t_2) = \varnothing$. Consequently, we have $u_1 \neq u_2$, $\mathrm{var}(u_1) \cap \mathrm{var}(u_2) = \varnothing$, and $Q_1 \cap Q_2 = \varnothing$. This implies that there exist two transitions such that:

$$\langle (G \setminus \{t_1\}) \cup \tau_1(Q_1), R \cup \{(\#t_1, \#u_1)\}, \tau_1 \circ \sigma \rangle \xrightarrow{N,P}$$
$$\langle (G \setminus \{t_1, t_2\}) \cup \tau_1(Q_1) \cup \tau_2(Q_2), R \cup \{(\#t_1, \#u_1), (\#t_2, \#u_2)\}, (\tau_1 \cup \tau_2) \circ \sigma \rangle$$
$$\langle (G \setminus \{t_2\}) \cup \tau_2(Q_2), R \cup \{(\#t_2, \#u_2)\}, \tau_2 \circ \sigma \rangle \xrightarrow{N,P}$$
$$\langle (G \setminus \{t_1, t_2\}) \cup \tau_1(Q_1) \cup \tau_2(Q_2), R \cup \{(\#t_1, \#u_1), (\#t_2, \#u_2)\}, (\tau_1 \cup \tau_2) \circ \sigma \rangle$$

Consequently, there is no source of non-determinism in the choice of the positive ground term according to which the transition is done. This means that

the search for a successful sequence of transitions may be organised as an and/or-tree. In Appendix A, such an and/or-tree is given for the following associative PN-matching problem:

$$P = \{1 : abc, 2 : Z, 3 : Y, 4 : eX, 5 : Wd\} \quad \text{and}$$
$$N = \{1' : c, 2' : YbZ, 3' : e, 4' : d, 5' : aXW\}$$

The main nodes, in this tree, are labelled with ground terms and the edges growing from these correspond to the different negative terms that match with the ground terms labelling the main nodes. Each such edge is labelled with the index of the corresponding negative term. Then there is a possible or-node with leaving edges corresponding to the possible different unifiers. Finally, each possible unifier gives rise to a and-node whose leaving edges reach the new positive ground terms resulting from a transition.

In such an and/or-tree, the subtree growing out of a main node is history independent: it is completely determined by the ground term labelling the main node. Consequently, the proof-search space may be organised as a DAG rather than as a tree (by using memoization or dynamic programming techniques, for instance).

## 6 Conclusions and future work

We have reduced **INL**, **IL**, and **IMLL** provability to matching problems that emphasise the part played by associativity in the case of **IL**, and associativity and commutativity in the case of **IMLL**. As we said in the introduction, we hope that this reduction will give an insight into the complexity of the Lambek calculus. An important question, in this context, is to know wether our PN-matching algorithm runs in polynomial time in the non-associative case. In fact, even in this simple case, it is not difficult to construct families of formulas for which the proof-search space, when organised as a tree, has an exponential number of nodes (see Appendix B). Consequently, the only hope of obtaining a polynomial algorithm is to organise some sharing as we suggested at the end of the previous section. If we do so, our PN-matching algorithm runs in polynomial time provided that the number of different positive ground terms involved in the search is polynomial. Several experimental results (see Appendix B, for instances) suggest that this is the case. Unfortunately, we do not know how to prove it in general. Therefore, the next step of this work will be to solve this question.

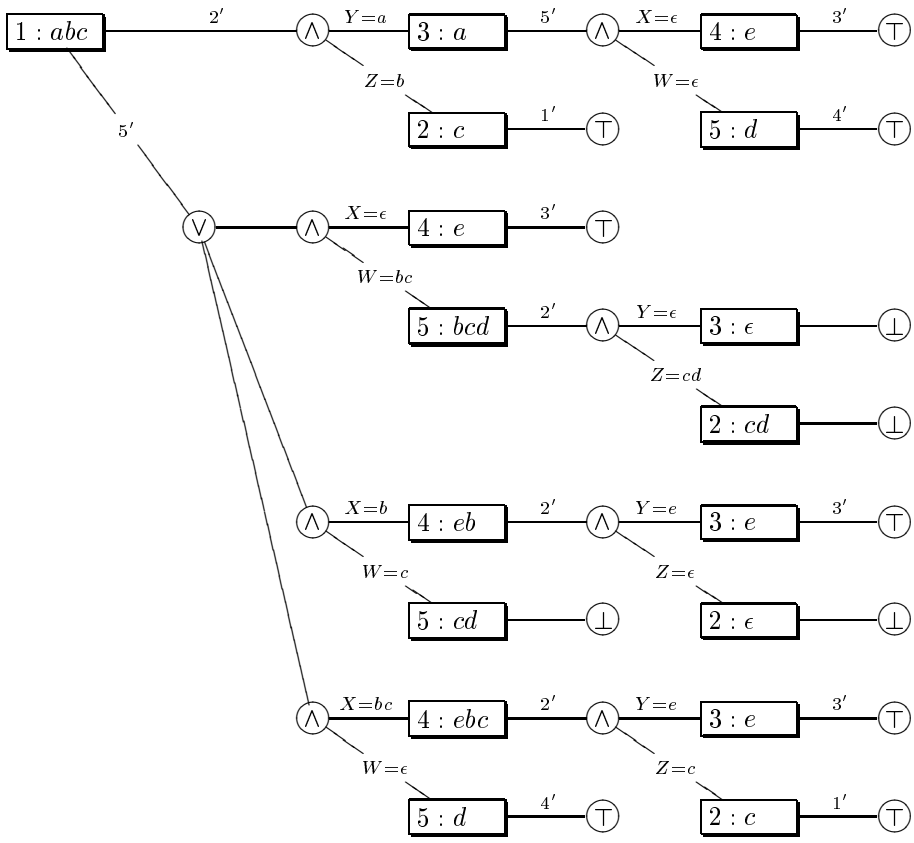
The first experiments we have conducted seem to indicate that our PN-matching algorithm has a good behaviour in practice. Hence, it would be interesting to see how our approach compete with other methods [16,23]. In this practical setting, working only in the implicative fragment of mutiplicative linear logic is a limitation. This raised the question of extending our procedure to fragments including additives and exponentials. In this respect, [11] might be a source of inspiration. Indeed, in the purely implicative case, there is a strong connection between our proof-search algorithm and Lamarche's games.

## References


1. E. Aarts and K. Trautwein. Non-associative Lambek categorial grammar in polynomial time. *Mathematical Logic Quaterly*, 41:476–484, 1995.
2. C. Berge. *Graphs*. North-Holland, second revised edition edition, 1985.
3. V. Danos. *Une application de la logique linéaire à l'étude des processus de normalisation et principalement du lambda calcul*. Thèse de doctorat, Université de Paris VII, 1990.
4. V. Danos and L. Regnier. The structure of multiplicatives. *Archive for Mathematical Logic*, 28:181–203, 1989.
5. Ph. de Groote. An algebraic correctness criterion for intuitionistic multiplicative proofnets. *Theoretical Computer Science*, 224:115–134, 1999.
6. Ph. de Groote. The non-associative lambek calculus with product in polynomial time. In *International Conference on Theorem Proving with Analytic Tableaux and Related Methods*, volume 1617 of *Lecture Notes in Artificial Intelligence*, pages 128–139. Springer Verlag, 1999.
7. J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.
8. S. Guerrini. Correctness of multiplicative proof-nets is linear. In *Proceedings of the fourteenth annual IEEE symposium on logic in computer science*, pages 454–463, 1999.
9. M. Kanovich. Horn programming in linear logic is np-complete. In *7-th annual IEEE Symposium on Logic in Computer Science*, pages 200–210. IEEE Computer Society Press, 1992.
10. Yves Lafont. From proof nets to interaction nets. In J.-Y. Girard, Y. Lafont, and L. Regnier, editors, *Advances in Linear Logic*, pages 225–247. Cambridge University Press, 1995.
11. F. Lamarche. Games semantics for full propositional linear logic. In *Ninth Annual IEEE Symposium on Logic in Computer Science*. IEEE Press, 1995.
12. F. Lamarche and C. Retoré. Proof nets for the Lambek calculus. In M. Abrusci and C. Casadio, editors, *Proofs and Linguistic Categories, Proceedings 1996 Roma Workshop*. Cooperativa Libraria Universitaria Editrice Bologna, 1996.
13. J. Lambek. The mathematics of sentence structure. *Amer. Math. Monthly*, 65:154–170, 1958.
14. J. Lambek. On the calculus of syntactic types. In *Studies of Language and its Mathematical Aspects*, pages 166–178, Providence, 1961. Proc. of the 12th Symp. Appl. Math..
15. P. Lincoln and T. Winkler. Constant-only multiplicative linear logic is NP-complete. *Theoretical Computer Science*, 135:155–169, 1994.
16. H. Mantel and J. Otten. linTAP: A tableau pover for linear logic. In *International Conference on Theorem Proving with Analytic Tableaux and Related Methods*, volume 1617 of *Lecture Notes in Artificial Intelligence*, pages 217–231. Springer Verlag, 1999.
17. M. Moortgat. Categorial type logic. In J. van Benthem and A. ter Meulen, editors, *Handbook of Logic and Language*, chapter 2. Elsevier, 1997.
18. G. Morrill. *Type Logical Grammar: Categorial Logic of Signs*. Kluwer Academic Publishers, Dordrecht, 1994.
19. M. Okada. A graph-theoretic characterization theorem for multiplicative fragment of non-commutative linear logic. *Electronic Notes on Theoretical Computer Science*, 3, 1996.

20. M. Pentus. Language completeness of the Lambek calculus. In *Proceedings of the ninth annual IEEE symposium on logic in computer science*, pages 487–496, 1994.
21. C. Retoré. Calcul de Lambek et logique linéaire. *Traitement Automatique des Langues*, 37(2):39–70, 1997.
22. D. Roorda. *Resource Logics: proof-theoretical investigations*. PhD thesis, University of Amsterdam, 1991.
23. T. Tammet. Proof strategies in linear logic. *Journal of Automated Reasoning*, 12:273–304, 1994.
24. J. van Benthem. *Language in action: Categories, Lambdas and Dynamic Logic*, volume 130 of *Sudies in Logic and the foundation of mathematics*. North-Holland, Amsterdam, 1991.

## A An example of proof-search space



## B Experimental results

The experiments we report here concern only the non-associative case. They have been conducted using a CAML light programme of almost two hundred lines.

A first bench mark consists in the following family of formulas.

$$\begin{aligned} f_0 &= A \\ f_{i+1} &= (f_i \circ\!\!- (f_i \multimap f_i)) \multimap f_i \\ F_i &= f_i \multimap f_i \end{aligned}$$

This family has the property that each formula $F_n$ has a number of different proofs that is exponential in the length of $F_n$. In fact, the length of $F_n$ is $2 \cdot 4^n$ and its number of different proofs is a double exponential in $n$, namely:

$$3^{\sum_{i=0}^{n-1} 4^i}.$$

Our programme does not search for only one possible proof but computes the number of different proofs. Consequently, it explores the entire proof-search space. The results of our experiment are given in the following table. The second column gives the length of the formula. The third column gives the number of nodes in the proof-search space when it is organised as a tree. On the other hand, the fourth column gives the number of different nodes, i.e, the number of nodes in the proof-search space when it is organised as a DAG. For $n > 3$, the algorithm without sharing is not feasible. This explains the absence of result in the third column.

| $n$ | length of $F_n$ | number of nodes without sharing | number of nodes with sharing |
|---|---|---|---|
| 1 | 8 | 9 | 6 |
| 2 | 32 | 390 | 42 |
| 3 | 128 | 302,549 | 294 |
| 4 | 512 | — | 1,968 |
| 5 | 2048 | — | 12,696 |

A second bench mark is the following.

$$\begin{aligned} g_0 &= A \\ g_{i+1} &= ((A \multimap A) \circ\!\!- A) \multimap g_i \\ G_i &= g_i \multimap f_i \end{aligned}$$

Each formula of this family has only one proof. However, the search for this proof gives rise to a number of failures that grows exponentially when there is no sharing.

| $n$ | length of $G_n$ | number of nodes without sharing | number of nodes with sharing |
|---|---|---|---|
| 1 | 8 | 7 | 7 |
| 2 | 14 | 27 | 14 |
| 3 | 20 | 86 | 22 |
| 4 | 26 | 271 | 31 |
| 5 | 32 | 838 | 41 |
| 6 | 38 | 2,555 | 52 |
| 7 | 44 | 7,679 | 64 |
| 8 | 50 | 22,777 | 77 |
| 9 | 56 | 66,768 | 91 |
| 10 | 62 | 193,719 | 106 |