Constructing different phonological bracketings from a proof-net

Denis Bechet and Philippe de Groote

Projet Calligramme
INRIA-Lorraine – CRIN – CNRS
615, rue du Jardin Botanique - B.P. 101
54602 Villers-lès-Nancy Cedex – FRANCE
e-mail: bechet@loria.fr, degroote@loria.fr

Abstract. We state and prove Roorda's interpolation theorem in the framework of proof-net theory. This allows us to transform any proof-net into some other proof-net that matches some given (phonological or prosodic) bracketing.

1 Introduction

Almost a decade ago, Girard invented linear logic together with the notion of proof-net [7]. Girard's proof-nets have been subsequently adapted to the Lambek calculus by Roorda [16] and, since then, many authors have advocated the notion of proof-net as the right parsing structure in the framework of categorial grammars [11, 13, 14, 16]. Nevertheless, if one wants to take this proposal seriously, one must be able to perform, on the proof-nets, all the computations that one usually performs on Gentzen's sequential derivations.

From a theoretical standpoint, the above possible objection is actually not a problem. Indeed, by Girard's sequentialisation theorem, one may always associate to any proof-net some corresponding sequential derivation. Therefore, any piece of information that can be computed from a sequential derivation may be computed from a proof-net. From a methodological and algorithmic point of view, however, one does not want to make the *détour* of sequentialising the proof-nets: one wants to compute information directly from the proof-nets.

In a recent paper [5], Retoré and one of the authors of the present paper have shown how to get semantic readings directly from proof-nets. In the same spirit, we show, in this paper, how to assign a phonological term to any proof-net (using Moortgat's phonological algebra), and how to transform a given proof-net in order to get different phonological bracketings. Technically, the main difficulty consists of stating and proving, in the framework of the proof-nets, Roorda's interpolation theorem [16]. This problem is solved in Section 7.

The paper is organised as follows. The next section is a short review of the Lambek calculus. Section 3 introduces Moortgat's phonological algebra together with a sequent calculus that assigns some bracketing to any derivation of a given sequent. Section 4 shows how to transform a derivation in order to stick to some given bracketing. Section 5 is a brief introduction to proof-net theory. Section 6

explains how to read the phonological bracketing associated with a proof directly from the corresponding proof-net. Finally, in Section 7, we state and prove our main technical result, i.e., Roorda's interpolation theorem in the framework of proof-net theory.

2 The Lambek calculus

The Lambek calculus, introduced as a logical basis for categorial grammars almost four decades ago [9], may be defined as the non-commutative intuitionistic multiplicative fragment of linear logic.[1]

The formulas (or types) of the Lambek calculus are built from an alphabet of atomic formulas ($\mathcal{A}$) according to the following grammar:

$$\mathcal{F} \ ::= \ \mathcal{A} \mid \mathcal{F} \bullet \mathcal{F} \mid \mathcal{F} \setminus \mathcal{F} \mid \mathcal{F}/\mathcal{F}$$

where formulas of the form $A \bullet B$ correspond to conjunctions (or products), formulas of the form $A \setminus B$ correspond to direct implications (i.e., *A implies B*), and formulas of the form A/B to retro-implications (i.e., *A is implied by B*).

The deduction relation of the calculus is defined by means of the following system:

Identity rules

$$A \vdash A \quad \text{(ident)} \qquad \frac{\Gamma \vdash A \quad \Delta_1, A, \Delta_2 \vdash B}{\Delta_1, \Gamma, \Delta_2 \vdash B} \quad \text{(cut)}$$

Logical rules

$$\frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, A \bullet B, \Delta \vdash C} \quad (\bullet \text{ left}) \qquad \frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma, \Delta \vdash A \bullet B} \quad (\bullet \text{ right})$$

$$\frac{\Gamma \vdash A \quad \Delta_1, B, \Delta_2 \vdash C}{\Delta_1, \Gamma, A \setminus B, \Delta_2 \vdash C} \quad (\setminus \text{ left}) \qquad \frac{A, \Gamma \vdash B}{\Gamma \vdash A \setminus B} \quad (\setminus \text{ right})$$

$$\frac{\Gamma \vdash A \quad \Delta_1, B, \Delta_2 \vdash C}{\Delta_1, B/A, \Gamma, \Delta_2 \vdash C} \quad (/ \text{ left}) \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash B/A} \quad (/ \text{ right})$$

where Greek uppercases range over sequences of formulas.

It is to be noted that the above system does not include any structural rule. In particular, the absence of the exchange rule is responsible for the non-commutativity of the connectives. This, in turn, explains the presence of two different implications.

[1] This is not entirely true because empty sequences of hypotheses are not allowed in the sequents of the original Lambek calculus. We do not insist on this restriction since the results presented in this paper hold in both system (i.e., the Lambek calculus with or without the empty sequence).

3 Moortgat's phonological algebra

Following Moortgat [12], one defines the phonological algebra associated with a set of tokens $\mathcal{V}$ as the structure freely generated from $\mathcal{V}$ by a binary operation "+", and one adds to this structure an identity element ε such that for any $a \in \mathcal{V}$, $(a + \varepsilon) = a$ and $(\varepsilon + a) = a$. Notice that, apart from these identity laws, the phonological algebra does not obey any law. In particular, "+" is not associative.

The purpose of the phonological algebra is to associate with any Gentzen derivation a phonological term that would reflect some prosodic phrasing [10, 12]. This may be achieved by decorating sequents with phonological terms:

$$a : A \vdash a : A \quad \text{(Id)} \qquad \frac{\Gamma \vdash t : A \qquad \Delta, a : A, \Theta \vdash u : B}{\Delta, \Gamma, \Theta \vdash u[a{:=}t] : B} \quad \text{(Cut)}$$

$$\frac{\Gamma \vdash t : A \qquad \Delta, b : B, \Theta \vdash u : C}{\Delta, \Gamma, a : A \setminus B, \Theta \vdash u[b{:=}(t + a)] : C} \quad (\setminus\text{L}) \qquad \frac{a : A, \Gamma \vdash t : B}{\Gamma \vdash t[a{:=}\varepsilon] : A \setminus B} \quad (\setminus\text{R})$$

$$\frac{\Gamma \vdash t : A \qquad \Delta, b : B, \Theta \vdash u : C}{\Delta, a : B/A, \Gamma, \Theta \vdash u[b{:=}(a + t)] : C} \quad (/\text{L}) \qquad \frac{\Gamma, a : A \vdash t : B}{\Gamma \vdash t[a{:=}\varepsilon] : B/A} \quad (/\text{R})$$

where a and b range over tokens; t and u range over the terms of the phonological algebra; $t[a{:=}u]$ denotes the term obtained by substituting u for a, in t; in Rules (\R) and (/R), the token a is fresh.

4 Rebracketing by interpolation

Because of Rule (Cut), the phonological bracketing associated to a given derivation is not invariant under cut-elimination. This is due to the following fact: the Lambek calculus is associative whereas the phonological algebra is not. This apparent mismatch is actually not a drawback but entails the property of *structural completeness* [2], which may be used, for instance, to settle conflicts between intonational and syntactic phrasing.

Example 4.1 Syntactic phrasing: (pierre + (écoute + (marie + chanter))) (1)

$$\dfrac{\mathrm{m}{:}SN \vdash \mathrm{m}{:}SN \qquad \dfrac{x{:}SV \vdash x{:}SV \qquad \dfrac{\mathrm{p}{:}SN \vdash \mathrm{p}{:}SN \qquad z{:}P \vdash z{:}P}{\mathrm{p}{:}SN, y{:}SN\backslash P \vdash (\mathrm{p}{+}y){:}P}}{\mathrm{p}{:}SN, \mathrm{e}{:}(SN\backslash P)/SV, x{:}SV \vdash (\mathrm{p}{+}(\mathrm{e}{+}x)){:}P}}{\mathrm{p}{:}SN, \mathrm{e}{:}(SN\backslash P)/SV, \mathrm{m}{:}SN, \mathrm{c}{:}SN\backslash SV \vdash (\mathrm{p}{+}(\mathrm{e}{+}(\mathrm{m}{+}\mathrm{c}))){:}P}$$

Intonational phrasing: ((pierre + écoute) + (marie + chanter)) (2)

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\mathrm{p}{:}SN \vdash \mathrm{p}{:}SN} \quad \dfrac{}{w{:}P \vdash w{:}P}}{y{:}SV \vdash y{:}SV \quad \mathrm{p}{:}SN, z{:}SN\backslash P \vdash (\mathrm{p}{+}z){:}P} \quad \dfrac{\dfrac{}{y{:}SV \vdash y{:}SV} \quad \dfrac{}{z{:}P \vdash z{:}P}}{\mathrm{m}{:}SN \vdash \mathrm{m}{:}SN \quad x{:}P/SV, y{:}SV \vdash (x{+}y){:}P}}{\mathrm{p}{:}SN, \mathrm{e}{:}(SN\backslash P)/SV, y{:}SV \vdash (\mathrm{p}{+}(\mathrm{e}{+}y)){:}P} \quad x{:}P/SV, \mathrm{m}{:}SN, \mathrm{c}{:}SN\backslash SV \vdash (x{+}(\mathrm{m}{+}\mathrm{c})){:}P}{\mathrm{p}{:}SN, \mathrm{e}{:}(SN\backslash P)/SV \vdash (\mathrm{p}{+}\mathrm{e}){:}P/SV \quad x{:}P/SV, \mathrm{m}{:}SN, \mathrm{c}{:}SN\backslash SV \vdash (x{+}(\mathrm{m}{+}\mathrm{c})){:}P}}{\mathrm{p}{:}SN, \mathrm{e}{:}(SN\backslash P)/SV, \mathrm{m}{:}SN, \mathrm{c}{:}SN\backslash SV \vdash ((\mathrm{p}{+}\mathrm{e}){+}(\mathrm{m}{+}\mathrm{c})){:}P}\ (\mathrm{cut})$$

The difficulty in finding a derivation that corresponds to a given prosodic phrasing such as (2) consists in guessing the needed *cut formulas*—P/SV, in our simple example—[10]. When working in the Lambek calculus with product, the problem of guessing the cut formulas may be circumvented by using Roorda's interpolation theorem [16]. This lemma, however, is stated in the framework of the sequent calculus. Therefore, if one wants to stick to the formalism of proof-nets, one has to answer the two following questions:

- how to assign phonological terms to proof-nets;
- how to compute interpolants from proof-nets.

Sections 6 and 7 are devoted to these two questions while the next section is a crash review of proof-net theory.

5 Proof-nets for the Lambek calculus

Proof-nets, which have been introduced by Girard [7] as an appropriate way of representing proofs in linear logic, are defined as a special class of graph whose nodes are decorated with formulas.

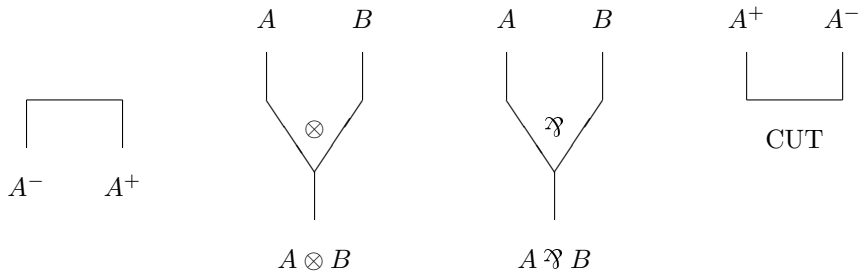
In this section, we review the usual notion of multiplicative proof-net (i.e., the notion of proof-net that fits classical multiplicative linear logic). Then we explain briefly how this notion may be adapted to the Lambek calculus (i.e., how to take *intuitionism* and *non-commutativity* into account). The reader who would like to know more details on the subject is referred to [15, 16].

We first introduce the formulas of multiplicative linear logic. They obey the following grammar:

$$\mathcal{F} \ ::= \ \mathcal{A}^+ \ \mid \ \mathcal{A}^- \ \mid \ \mathcal{F} \otimes \mathcal{F} \ \mid \ \mathcal{F} \parr \mathcal{F}$$

Where the connectives "$\otimes$" and "$\parr$" are respectively called *tensor* and *par*.

Then, we consider the following *links* that are respectively called *axiom*, $\otimes$*-link*, $\parr$*-link*, and *cut*:



where

- the formulas A^- and A^+ are defined to be the conclusions of the *axiom*;
- the formula $A \otimes B$ is defined to be the conclusion of the $\otimes$-*link* while the formulas A and B are defined to be its left and right premises;
- the formula $A \parr B$ is defined to be the conclusion of the $\parr$-*link* while the formulas A and B are defined to be its left and right premises;
- the formulas A^+ and A^- are defined to be the premises of the *cut*;
- when A is not an atomic formula, the formula A^+ is simply defined to be A itself while the formula A^- is defines according to the following inductive clauses:

$$(A \otimes B)^- = A^- \parr B^-$$
$$(A \parr B)^- = A^- \otimes B^-$$

Note that we distinguish between the left and the right premise only for the $\otimes$-link and the $\parr$-link. There is no notion of left and right premise for a cut. Similarly, There is no notion of left and right conclusion for an axiom.

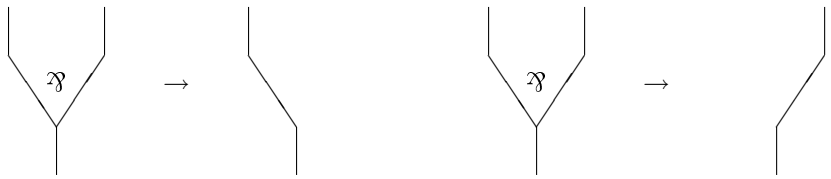
The notion of link allows the notion of *proof-structure* to be defined.

A proof-structure is a set of (occurrences of) formulas connected by links, such that every (occurrence of a) formula is a conclusion of exactly one link and is a premise of at most one link. The (occurrences of) formulas that are not the premise of any link are called the conclusions of the proof-structure.

Given a proof-structure, the graph obtained by removing all the axioms is called a *proof-frame*. In fact such a proof-frame is nothing but a forest of syntactic trees of formulas.

Proofs in linear logic will be represented by proof-structures. It is not the case, however, that every proof-structure corresponds to some actual proof in the sequent calculus. In fact, the proof-structures that correspond to actual proofs are defined to be the proof-nets. Now, the keystone of proof-net theory is that these proof-nets may be globally characterised by giving some correctness criterion that allows them to be discriminated from the other proof-structures (therefore the notion of proof-net may be defined without making any explicit reference to the sequent calculus). In order to introduce such a correctness criterion, which is due to Danos and Regnier [4], we must define the notion of *switching*.

A switching of a proof-structure is a selection for every $\parr$-link between the *left* or the *right* position. The graph underlying such a switching is obtained by replacing each $\parr$-link by a single edge as follows:



left position right position

We are now in a position to define formally the notion of multiplicative proof-net: *a proof-structure is a proof-net if and only if for every possible switching the underlying graph is connected and acyclic.*

As we said, we must take intuitionism and non-commutativity into account in order to accommodate the above notion of proof-net to the Lambek calculus. To this end, the following positive and negative translations are introduced:

1. $(A)^+ = A^+$, if A^+ is atomic,
2. $(A \setminus B)^+ = B^+ \parr A^-$,
3. $(A/B)^+ = B^- \parr A^+$,
4. $(A \bullet B)^+ = B^+ \otimes A^+$,
5. $(A)^- = A^-$, if A^+ is atomic,
6. $(A \setminus B)^- = A^+ \otimes B^-$,
7. $(A/B)^- = A^- \otimes B^+$,
8. $(A \bullet B)^- = A^- \parr B^-$.

These translations allow us to transform any intuitionistic sequent $A_0, \cdots, A_n \vdash B$ of the Lambek calculus into a sequence $A_0^-, \cdots, A_n^-, B^+$ of formulas of multiplicative linear logic. Then, to allow for non-commutativity amounts to specifying some planarity constraints. This leads to the following definition.

A *non-commutative intuitionistic multiplicative proof-net* associated to a sequent $A_0, \cdots, A_n \vdash B$ is a multiplicative proof-net whose conclusions are $A_0^-, \cdots, A_n^-, B^+$ and that may be represented by a topological planar graph [1, p. 16] such that:

1. the topological planar representation respects the notion of left and right premise;
2. all the conclusions appear on the external boundary of the representation;
3. when following the external boundary counterclockwise one meets the conclusions in the order $A_0^-, \cdots, A_n^-, B^+$ (up to a circular permutation);
4. the contour of each (bounded) face contains exactly one $\parr$-link (this last condition, which is redundant for the cut-free proof-nets, is mandatory in the presence of cuts).

Note that such proof-nets are intuitionistic because they have only one positive conclusion (B^+). This conclusion will be called the output conclusion while the other ones ($A_0^-, \cdots, A_n^-$) will be called the input conclusions.

In fact we may incorporate the above positive and negative translations within the proof-nets in order to make explicit the notion of link for the Lambek calculus. This leads to the $\parr$-links and $\otimes$-links given by Fig. 1. The signs (+

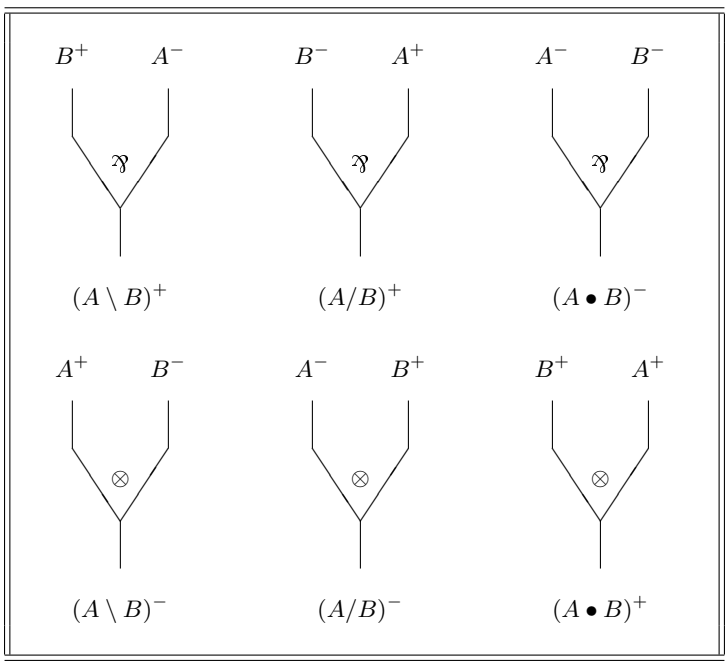


Fig. 1. $\parr$-links and $\otimes$-links for the Lambek calculus

and $-$) that appear at the level of the premises and conclusions of these links are called *positive* and *negative polarities* (or, respectively, *output* and *input polarities*). Note that the polarities assigned to the premises of a link determine unequivocally the polarity assigned to its conclusion. A priori, there are four different ways of assigning polarities to the two premises of a link. However, some of these possible configurations are forbidden, namely the $+/+$ assignment for the $\parr$-link and the $-/-$ assignment for the $\otimes$-link. This is due to the intuitionistic nature of the Lambek calculus.

From now on, we will say proof-net for non-commutative intuitionistic multiplicative proof-net.

The above notion of proof-net satisfies the following property that justifies that non-commutative intuitionistic multiplicative proof-nets correspond to the proof-nets of the Lambek calculus.

Proposition 5.1 *A sequent $\Gamma \vdash B$ is provable in the Lambek calculus if and only if there exists an associated proof-net.* □

The only-if-part of this proposition may be easily established by induction

on the derivations of the sequent calculus. The if-part, which corresponds to Girard's sequentialisation theorem, can be proven using the *splitting $\parr$-link method* due to Danos [3].

A $\parr$-link occurring in a proof-net is called *splitting* if its removal splits the proof-net in two parts, one connected to the two premises and one to the conclusion of the link (see Fig. 2).

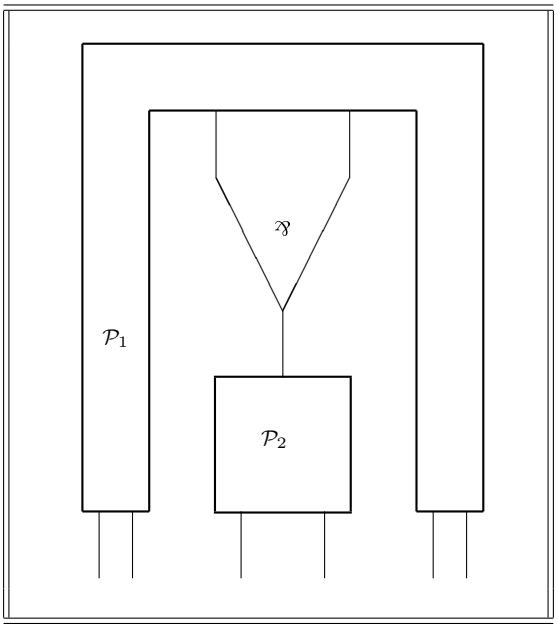


Fig. 2. A splitting $\parr$

The splitting-$\parr$ method is based on the following lemma.

Lemma 5.2 *Any proof-net that contains at least one $\parr$-link contains a splitting $\parr$-link on the external boundary of its topological planar representation.*

This lemma is proven in [3] for classical multiplicative linear logic. The proof is similar in the intuitionistic, non-commutative case. The only novelty, which is due to our non-commutative setting, is that the splitting $\parr$ must be on the external boundary of the proof-net [6].

Given a splitting $\parr$ (see Fig. 2), the two parts $\mathcal{P}_1$ and $\mathcal{P}_2$ induce two smaller proof-nets. The part connected to the two premises (i.e., $\mathcal{P}_1$) is directly a proof-net. The part connected to the conclusion (i.e., $\mathcal{P}_2$) with an additional axiom is a proof-net. This fact allows the sequentialisation theorem to be proven by induction on the size of the proof-nets.

6 Reading the phonological bracketing from a proof-net

In [16, pp. 39-40], Roorda defines a procedure that assigns a string to each node of a proof-net. This procedure may be adapted (actually simplified) in order to compute the phonological term associated to a given proof-net.

Technically, we work with the phonological algebra generated by a set $\mathcal{V} = \mathcal{P} \cup \mathcal{X}$, where $\mathcal{P}$ and $\mathcal{X}$ are two disjoint sets, respectively called the set of parameters and the set of variables. The terms that do not contain variables will be called proper terms.

The computation of the proper phonological term associated to a proof-net goes as follows:

1. Assign to each input conclusion (negative conclusion) of the proof-net a different parameter (In practice, the atomic constituents of the phrase); assign to the output conclusion (positive conclusion) of the proof-net a variable (say x).
2. Assign to each node of the proof-net a term (or a variable) according to the unfolding described in Fig. 3.
3. Consider the unification problem made of the constraints associated to some of the links, and of the equations "*variable* = *term*" resulting from the axiomatic links. By solving this unification problem, one finds a proper term for the variable x. This term is the phonological term associated to the proof-net.

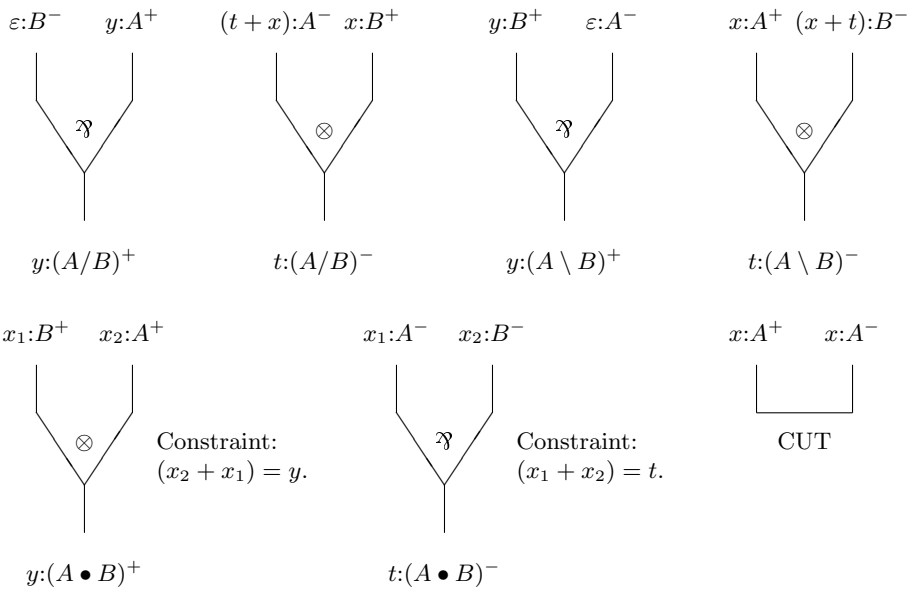
Roorda's results [16] ensure that the unification problem involved in the computation of a phonological term always admits a solution. Moreover, it is easy to show that the phonological term associated to a proof-net (according to our procedure) is the same as the one that would be obtained by considering some corresponding sequential proof. Notice that the unfolding procedure that we give is actually a simplification of Roorda's: in particular, associative unification is not needed.

7 Interpolation on proof-nets

Roughly speaking, Roorda's interpolation theorem says that whenever a sequent Γ, Ψ, $\Delta \vdash A$ is provable, there exists a formula I (called an interpolant) such that the two sequents $\Psi \vdash I$ and Γ, I, $\Delta \vdash A$ are provable. In addition, the atomic subformulas of I must obey some occurrence conditions (see [16] for details).

Now consider a proof-net, say Π, with n input conclusions C_i^- $(1 \leq i \leq n)$ and one output conclusion C^+. In order to state an interpolation problem for this proof-net, one must distinguish some consecutive input conclusions, say $C_j^-, C_{j+1}^-, \ldots, C_{j+k}^-$. Then a solution to this interpolation problem consists of one formula I and two proof-nets Π_1 and Π_2 such that:

1. the input conclusions of Π_1 are $C_j^-, C_{j+1}^-, \ldots, C_{j+k}^-$, and its output conclusion is I^+;



where t is a phonological term; x, x_1, x_2, y range over variables; for each link, the variables x, x_1, x_2 are chosen to be fresh.

Fig. 3. unfolding

2. the input conclusions of Π_2 are $C_1^-, C_{j-1}^-, I^-, C_{j+k+1}^-, \ldots, C_n^-$, and its output conclusion is C;
3. in Π_1 (resp. Π_2), there is no axiomatic link both conclusions of which would belong to the proof-frame associated to I^+ (resp. I^-);
4. in Π_1 (resp. Π_2), all the axiomatic links no conclusion of which belong to the proof-frame associated to I^+ (resp. I^-) were already existing in Π.

More abstractly, but equivalently, an interpolation problem consists of a proof-net Π together with a set of n axiomatic links whose removal splits the proof-net Π into two disconnected parts M_1 and M_2 (see Fig.4). These parts, which are not proof-nets, will be called modules. Without lost of generality, consider that the output conclusion of Π is a conclusion of M_2. Then, a solution to this interpolation problem consists of a formula I of length n such that the proof-structure obtained by

1. linking the module M_1 to the proof-frame I^+ using n axioms,
2. linking the conclusions of the proof-frames I^+ and I^- by a cut,
3. linking the proof-frame I^- to the module M_2 using n axioms,

is a proof-net (see Fig. 5).

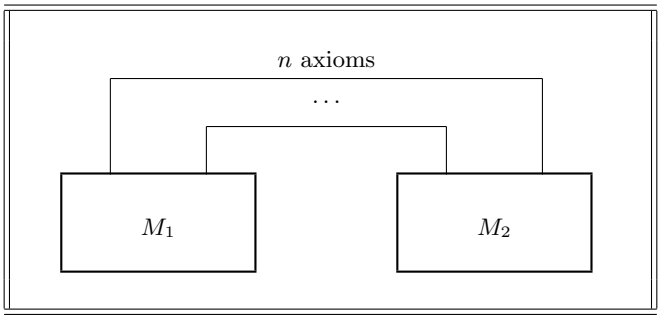


Fig. 4. Interpolation problem

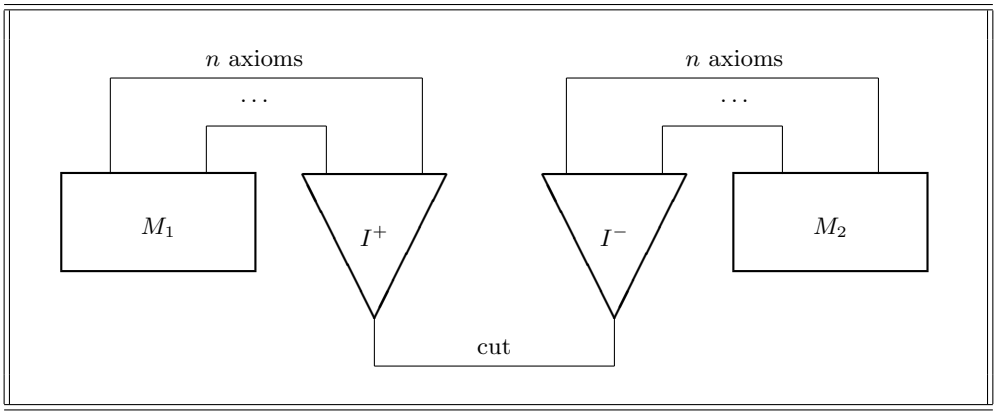


Fig. 5. Solution to the interpolation problem

In order to show that any interpolation problem admit a solution, we establish two key lemmas.

Lemma 7.1 *Consider a proof-net containing a set of n cuts ($n > 1$) whose removal splits the proof-net into two disconnected modules (see Fig.6). Then, there exist, among these n cuts, two cuts and one of the two modules M_1 or M_2 (say M_i) such that for any switching, the two corresponding conclusions are always connected by a path belonging to M_i. Moreover, one may always find two such cuts that are adjacent.*

Proof. The second part of the proposition, which concerns the fact that the cuts may be found adjacent, is rather involved and quite long because of numerous cases and subcases. The difficulty arises from the fact that one must use *planarity arguments* that cannot be easily formalised. For this reason, we reserve the complete proof for a long paper and establish only the first part of the proposition.[2]

[2] Consequently, in the present paper, our proof of the interpolation theorem is only complete for the (intuitionistic) commutative case.

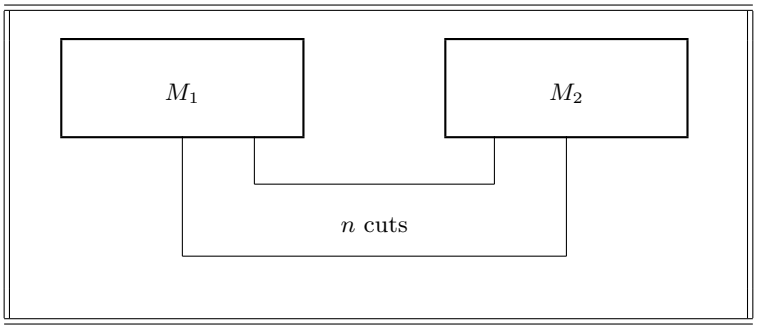


Fig. 6.

The proof is by induction on the number of ⅋-links in the proof-net.

If the proof-net does not contain any ⅋-link, it has only axioms, cuts and ⊗-links. Therefore, there is only one possible switching (the empty one). Then, because any proof-net is connected, there exist two cuts that are connected inside the same module (M_1 or M_2).

If the proof-net $\mathcal{P}$ has at least one ⅋-link, there exists, by Lemma 5.2, a splitting ⅋-link that induces two proof-nets $\mathcal{P}_1$ and $\mathcal{P}_2$. The n initial cuts are then divided in two sets A and B belonging respectively to the proof-nets $\mathcal{P}_1$ and $\mathcal{P}_2$ (see Fig. 7). There are three cases:

- The number of cuts in A is greater or equal to 2. Then we are done by applying the induction hypothesis to the proof-net $\mathcal{P}_1$.
- The number of cuts in B is greater or equal to 2. Then we apply the induction hypothesis on the proof-net $\mathcal{P}_2$.
- The numbers of cuts in A and B are exactly one in both cases. Then the two cuts may not be connected within the module that does not contain the splitting ⅋ (see Fig. 7). This implies that they must be connected, for any switching, by a path belonging to the other module. □

The above lemma, which provides the most important part of the interpolation theorem, yields the next lemma as an almost direct consequence.

Lemma 7.2 *Consider a proof-net containing a set of n cuts whose removal splits the proof-net into two disconnected modules (see Fig. 6). Then, there exist, among these n cuts, two adjacent cuts that may be replaced by a single cut between a "⊗" and a "⅋" in such a way that the resulting proof-structure is a proof-net (see Fig. 8).*

Proof. This proposition is a consequence of the previous lemma. For the configuration of Fig. 6, we can apply the previous lemma and find two adjacent cuts connected, for any switching, by a path belonging to one of the modules (say M_i). Now, we can add a ⅋-link on the side of M_i and a ⊗-link on the other side as is shown on Fig. 8.

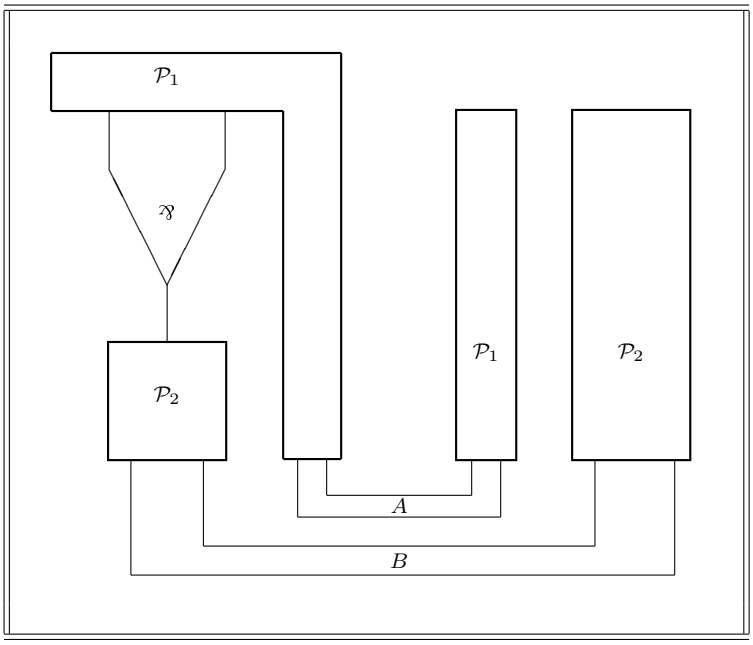


Fig. 7.

Now, it remains to prove: firstly, that this construction is always possible (it could be impossible for polarity reasons); secondly, that it yields a proof-net (i.e., a *correct* proof-structure).

Regarding the polarity problem, we must decide which sort of $\parr$-link we add: $+/-$, $-/+$, or $-/-$ (this determines the sort of $\otimes$-link to add, by duality). In fact, there is no degree of freedom here because we must respect the polarities of the initial proof-net. However, we must prove that the polarities of the premises of the new $\parr$-link are not both output (because that corresponds to the configuration $+/+$ which is forbidden in the intuitionistic proof-nets).

Consider a switching such that all the $+/-$ $\parr$-links are switched on the left and all the $-/+$ $\parr$-links are switched on the right. The graph underlying this switching is a tree whose root is the output conclusion of the original proof-net and whose edges (starting from the root) "go up" through positive formulas, cross the axioms in the output/input direction, "go down" through negative formulas, and cross the cuts in the input/output direction. Therefore, there exist two paths from the root of the tree to the two negative premises of the two cuts of interest. On the other hand, there also exists, for this switching, a path between the two premises of the $\parr$-link we are going to add. Therefore, if these two premises were positive, there would exist a path between the two positive premises of the two cuts. This would make a cycle, which is a contradiction.

It remains to prove that the construction is correct. Any switching of the new proof-structure is determined by a switching of the initial proof-net together

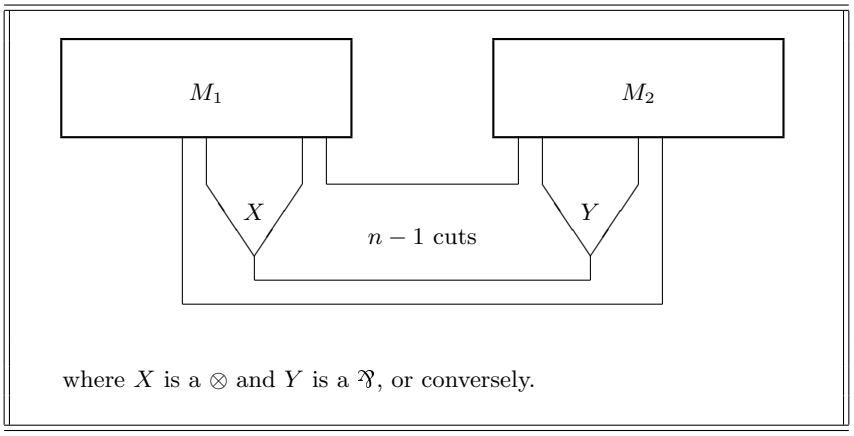


Fig. 8.

with a switch for the additional $\parr$-link. For the initial proof-net, we know that the graph has no cycle and is connected. We know that there exists a path between the two selected cuts in $\mathcal{M}_i$. This path remains in the new proof-structure. So the additional $\otimes$-link does not create a new cycle in the graph. For the same reason, because the initial proof-net is connected for any switching, the new one is also connected. Thus the new proof-structure is correct and is, by definition, a proof-net. □

We are now in a position to prove the interpolation theorem.

Theorem 7.3 *Any interpolation problem admits a solution.*

Proof. Given a proof-net such as the one in Fig. 4, one easily obtains a proof-net such as the one in Fig. 5 by replacing each of the n axioms by a path made of one axiom, one cut, and one axiom. Then, one may obtain an interpolant by iterating the key lemma. □

The proof of this theorem gives an algorithm to compute interpolants. The method is as follows:

1. Let n be the number of axioms that splits the given proof-net in two disconnected modules. Replace each of these axioms by a path *axiom-cut-axiom*.
2. If $n = 1$ then stop. (the interpolation problem is already solved)
3. Otherwise, find two adjacent cuts that obey the property described in Lemma 7.2. This corresponds to finding, in a module, two conclusions that are connected for any *switching* [4]. This last property is reminiscent of the notion of *empire* and may be checked using Girard's closure conditions on empires [8].
4. Replace the two adjacent cuts by a single cut as specified in Lemma 7.2.
5. Decrease n by 1 and go to step 2.

This algorithm, which works on proof-nets, does not use any sequentialisation of the proof-nets. Its complexity is at most $m * n^2$, where m is the size of the proof-net and n is the number of axioms that splits the proof-net in two parts. This cubic complexity corresponds actually to a rough estimate. Using tabularisation techniques, we could obtain a sub-quadratic algorithm.

8 Concluding remarks

We said, in Section 3, that the difficulty in finding a derivation corresponding to a given prosodic phrasing consists in guessing the cut formulas. In fact, this difficulty is not a technical one. Indeed, when working in the Lambek calculus with product, one may always group together two constituents of type A and B by assigning them the type $(A \bullet B)$. Such a trivial solution, of course, is not satisfactory.

The real problem in guessing the cut formulas is to provide the prosodic constituents with categorial types *that make sense*. With respect to this, the use of interpolation seems promising. Indeed, an interpolant may be interpreted as a type whose every atomic subtype specifies an actual interaction with the *external world*. This means, in our case, interactions between the constituent (to which the interpolant is assigned as a type) and the rest of the phrase.

Nevertheless, working with interpolants does not settle the problem completely because, in general, an interpolation problem admits several solutions. Therefore, what is now needed is a mathematical classification of these different solutions that would allow some canonical choice to be made.

References

1. C. Berge. *Graphs*. North-Holland, second revised edition edition, 1985.
2. W. Buszkowski. Generative power of categorial grammars. In R. T. Oehrle, E. Bach, and D. Wheeler, editors, *Categorial Grammars and Natural Languages Structures*, pages 69–94. Reidel, 1988.
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 and C. Retoré. On the semantic readings of proof-nets. In G.-J. Kruijff, G. Morrill, and D. Oehrle, editors, *Formal Grammar*, pages 57–70. Eighth European Summer School in Logic Language and Information, August 1996.
6. A. Fleury. Private communication.
7. J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.
8. J.-Y. Girard. Quantifiers in linear logic II. Technical Report 19, Equipe de Logique Mathématique, Université de Paris VII, 1991.
9. J. Lambek. The mathematics of sentence structure. *Amer. Math. Monthly*, 65:154–170, 1958.

10. A. Lecomte. Prosodie et stratégie de calcul. Communication à la journée ATALA "Interaction prosodie-syntaxe", Février 1996.
11. A. Lecomte and C. Retoré. Pomset logic as an alternative categorial grammar. In *Proceedings of the Conference of the European Summer School in Logic, Language and Information*, Barcelona, 1995.
12. M. Moortgat. *Categorial Investigations: logical and linguistic aspects of the lambek calculus*. Foris Publications, 1988.
13. M. Moortgat. Categorial type logic. In J. van Benthem and A. ter Meulen, editors, *Handbook of Logic and Language*, chapter two. Elsevier, to appear.
14. G. Morrill. Memoisation of categorial proof nets: parallelism in categorial processing. In V. M. Abrusci and C. Casadio, editors, *Proofs and Linguistic Categories, Proceedings 1996 Roma Workshop*. CLUEB, 1996.
15. C. Retoré. Calcul de lambek et logique linéaire. *Traitement Automatique de Langues*, 37(2):39–70, 1997.
16. D. Roorda. *Resource Logics: proof-theoretical investigations*. PhD thesis, University of Amsterdam, 1991.