Generation with Semantic Proof Nets

Sylvain Pogodalla

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Generation with Semantic Proof Nets

Sylvain Pogodalla

N° 3878

January 2000

THÈME 2

ISSN 0249-6399 ISRN INRIA/RR--3878--FR+ENG

LORRAINE

Generation with Semantic Proof Nets

Sylvain Pogodalla

Thème 2 — Génie logiciel
et calcul symbolique
Projet Calligramme

Rapport de recherche n° 3878 — January 2000 — 25 pages

Abstract: Categorial grammars and Lambek calculus found a nice embedding in Linear Logic, and a lot of work have presented proof nets uses for linguistic purposes, with a special look at proof nets for Lambek calculus. But they have mainly explored the syntactic capabilities of proof nets, describing parsing processes. We present here our vision of the generation process based on semantic proof nets. The main features of this proposal consist in the use of proof nets for lambek calculus but also for semantic recipes of lexical entries, so that no lambda-term unification occur with their limitations (undecidability from second order). Based on a graph calculus, the solutions of this process can be expressed as a matrix computation.

Key-words: Lambek Calculus, Generation, Linear Logic, Proof Nets

This report was written in 1999 while doing my thesis at Xerox Research Centre Europe, 6 chemin de Maupertuis, 38240 Meylan-FRANCE, +33 (0) 4 76 61 50 50 and is also available as Mltt technical report (mltt-036) at http://www.xrce.xerox.com/publis/mltt/mltt-036.ps

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,
615, rue du Jardin Botanique, BP 101, 54602 Villers-Lès-Nancy (France)
Téléphone : +33 3 83 59 30 00 — Télécopie : +33 3 83 27 83 19

Génération à l’aide de réseaux de preuve sémantiques

Résumé : Les grammaires catégorielles et le calcul de Lambek peuvent s’exprimer clairement dans le cadre de la logique linéaire, et de nombreux travaux ont présenté l’utilisation des réseaux de preuve pour des préoccupations linguistiques. Mais ce sont surtout les processus syntaxiques qui ont été étudiés, en particulier ceux de l’analyse. Nous présentons ici notre point de vue sur le processus de generation à l’aide de réseaux de preuve sémantiques. Les principales caractéristiques de cette proposition sont l’utilisation des réseaux de preuve à la fois pour le calcul de Lambek et la valeur sémantique des items lexicaux, de façon à ne pas utiliser l’unification de lambda-termes avec leurs limitations (indécidabilité de l’unification dès le deuxième ordre). Basées sur un calcul de graphes, les solutions de ce processus peuvent être exprimées par un calcul matriciel.

Mots-clés : Calcul de Lambek, Génération, Logique linéaire, Réseaux de preuve

Introduction

From the expression of categorial grammars [Ajdukiewicz35] and Lambek calculus [Lambek58] in the framework of Linear Logic [Girard87], a lot of work have presented proof nets uses for linguistic purposes, with a special look at proof nets for Lambek calculus [Roorda91, Lamarche et al.96]. But they have mainly explored the syntactic capabilities of proof nets, describing parsing processes.

We want to present here our vision of the generation process based on the proposal of [dG et al.96] for semantic proof nets. The main features of this paper consist in the use of proof nets for lambek calculus, of the Curry-Howard isomorphism [Howard80, Girard et al.88], of semantic proof nets with semantic expressions *à la* Montague [Montague74, Dowty et al.81], and in an algorithm for proof search with a target proof net. In this work, we do not consider the choice of lexical items from a given semantic expression the syntactic realization of which we want to generate, but rather the way we can associate given lexical entries to fit with the given semantic expression and generate a syntactically correct expression.

Few works on generation with proof nets exist. The main one certainly is [Merenciano et al.97]. But according to the authors, for decidability reasons in λ-terms unification, a lot of restrictions occur: λ-terms expressing the semantic form of lexical entries must be linear, belong to second order at most and contain at least one free variable.

As we work with graph and matrix computation instead of λ-terms unification, such limitations do not occur anymore. Even if for the moment we only consider linear terms for semantic recipes.

1 Multi Usage Proof Nets

1.1 Proof Nets

In this section, we give a very brief presentation of proof nets, mainly to introduce the notations we use. Articles we refer to are more complete on this subject.

Proof-nets in linear logic has become familiar [Girard87, Retore93]. They represent proofs in linear logic with more accuracy than sequential proofs: on one hand they are more compact, on the other hand they identify unessentially different sequential proofs (for instance in the order of introducing the rules).

From a one-sided sequent and a sequential proof of it, we obtain a proof net by unfolding every formula as a tree (whose nodes are the binary connectives and leaves are formulas, e.g. atomic ones) and linking together the atoms occurring in the same axiom rule of the sequent calculus.

But proof nets have a more intrinsic definition that prevent us to come back every time to sequential proofs. They can be defined as graphs with a certain property (i.e. verifying a correctness criterion) such that every proof net with this property corresponds to a sequential proof and such that every proof net built from a sequential proof has this property.

In this paper, we do not consider all the proof nets, but a part of multiplicative ones: those of the intuitionistic implicative linear logic. In this case, sequents are made of several antecedent formulas, but only one succedent formula. To deal with the intuitionistic notion with proof nets (since we consider one-sided sequents), we use the notion of polarities with the *input ($\bullet$: negative)* and the *output ($\circ$: positive)* [Danos90, Lamarche95] to decorate formulas. Positive ones correspond to succedent formulas and negative ones to antecedent formulas.

Given the links of table 1, we define *proof structures* as graphs made of these links such that:

1. any premise of any link is connected to exactly one conclusion of some other link;
2. any conclusion of any link is connected to at most one premise of some other link;
3. input (resp. output) premises are connected to input (resp. output) conclusions of the same type.

Note that the two links for the negative and positive implication correspond to the two connectives of the linear logic: Tensor and Par, so that we name these links after these latter connectives. But in the following, only the graphical forms of the links play a role.

Proof nets are proof structures with no alternate elementary cycle (alternate means a path using a blue/bold edge then a red/regular one etc. Elementary means a path not using twice the same vertex but the first and the last one). Every two vertices should also be linked with an alternate elementary path. Proof nets also have only one output.

Table 1: Links

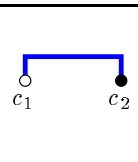
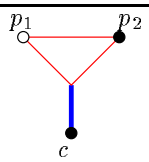
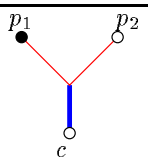
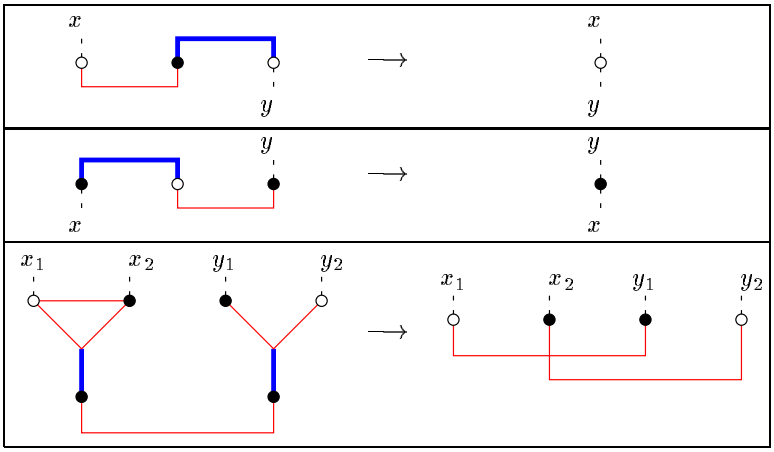
Name	*Axiom*	*Tensor*	*Par*	*Cut*
Link				
Premises	none	p_1, p_2	p_1, p_2	p_1, p_2
Conclusions	c_1, c_2	c	c	none
Types	$c_1 : A^+$ $c_2 : A^-$	$p_1 : A^+$ $p_2 : B^-$ $c : (A \multimap B)^-$	$p_1 : A^-$ $p_2 : B^+$ $c : (A \multimap B)^+$	$p_1 : A^-$ $p_2 : A^+$

Table 2: Cut elimination rewriting rules



We mentioned the intrinsic definition of proof nets that enables the complete representation of sequential proofs. The cut elimination property of sequent calculus also appears intrinsically in proof net formalism with a simple rewriting process described in table 2 (in case of complex formulas as in the third rewriting rule, those rules apply again on the result and propagate until reaching atoms).

1.2 Syntactic Proof Nets

Definitions of proof nets for Lambek calculus, first appear in [Roorda91]. They naturally raised as Lambek calculus is an intuitionistic fragment of non commutative linar logic (with two linear implications: "\" on the left and "/" on the right), and the consequences on the proof net calculus we presented in section 1.1 are:

- we get two tensor links: one for the formulas $(B/A)^-$ (the one in table 1) and one for the formula $(B\backslash A)^-$ (just inverse the polarities of the premises). And two par links : one for the formula $(A\backslash B)^+$ and one for $(A/B)^+$ (idem);
- formulas in Lambek's sequents are ordered, so that conclusions of the proof nets are cyclically ordered and axiom links may not cross.

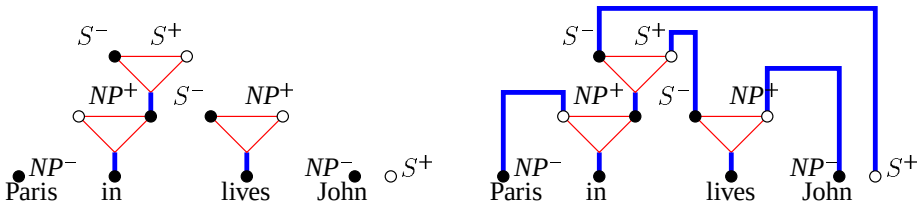
Note that from a syntactic category, we can unfold the formula to obtain a graph which only lacks axiom links to become a proof structure. So that the parsing process in this framework is, given the syntactic categories of the items

and their order, to put not crossing axiom links such that the proof structure is a proof net. It means there is a proof of S given types in a certain order. For technical reasons, the order of the conclusions (i.e. the types used) in the proof net to prove S is the reverse order of the words associated to these types.

As an example, with the lexicon of table 3, proving that *John lives in Paris* is a correct sentence leads to find axiom links between the atoms in the figure 1(a). Figure 1(b) shows it actually happens and proves the syntactic correctness of the sentence.

Table 3: Lexicon

lexical entry	syntaxic category
John	NP
Paris	NP
lives	$NP\backslash S$
in	$(S\backslash S)/NP$



(a) Unfolding of the syntactic types

(b) Matching the dual atoms to obtain a proof net

Figure 1: Parsing of *John lives in Paris*

1.3 Semantic Proof Nets

This section explains how we can use proof nets as the semantic readings associated to a lexical entry. It refers to the work of [dG et al.96] but restricted to the linear case. (The non linear case should be dealt with in further work. But conditions on proof search are such that it should keep a good part of the linear properties.)

The first idea uses the fact that both λ-terms (with the Curry-Howard isomorphism) and proof nets represent proofs of intuitionistic implicative linear logic. And indeed, the linear λ-terms may be encoded as proof nets. On the other hand, given an intuitionistic implicative proof net, a simple algorithm (given in [dG et al.96], based on [Lamarche95]'s dependency paths), we can obtain a λ-term.

This reading follows the algorithm:

1. enter the proof net by its unique output conclusion;
2. follow the output polarities until reaching an axiom link. This path is ascending and is made of par links corresponding to abstractions;
3. cross the axiom link to the input;
4. follow the input polarities. This path is descending and is made of tensor links corresponding to application of the conclusion to the positive premisse (that can be read as a λ-term with the same algorithm). If it ends on an input conclusion, this latter coincides with a free head- variable for the λ-term. If it ends on the premise of a par link, it coincides with a bound head-variable (the one bound to the λ corresponding to the par link).

Then, instead of associating a λ-term to a lexical entry, we can associate a proof net. For instance, on the semantic side, we can use the Montagovian types e and t and typed constants. To express the compositional principle between syntax and semantic, we use the following homomorphism:

$$\mathcal{H}(N) = e \multimap t \qquad \mathcal{H}(NP) = e \qquad \mathcal{H}(S) = t$$
$$\mathcal{H}(A\backslash B) = \mathcal{H}(A) \multimap \mathcal{H}(B) \qquad \mathcal{H}(A/B) = \mathcal{H}(B) \multimap \mathcal{H}(A)$$

And for a lexical item, given its syntactic type, we assume its semantic proof net to verify:

- the type of its unique output conclusion is the homomorphic image of the syntactic type;
- its input conclusions (if any) are decorated with typed constants.

An example of such a lexicon is given in table 4.

So that proof nets built on the syntactic analyses can (quite) directly provide the semantic analyses: the homomorphic image of the proof net resulting from syntactic analysis keeps the correction and expresses how the semantic recipes should be composed. Then to have the final (normal) form based on typed constant, we plug by means of cuts (i.e. we substitute) the inputs to their semantic recipes, and we eliminate these cuts (i.e. we reduce the term). The semantic reading of the whole sentence is the resulting proof net (which of course corresponds to a λ-term).

The section 2.1 illustrates the parsing process with an example.

2 Generation: Stating the Problem

In this section, we state the problem we want to solve, and we explain how solving this problem answers the generation problem. But to make things clearer, we present first the parsing process with syntactic and semantic proof nets, as proposed in [dG et al.96], on a simple example.

2.1 Parsing

Let us consider the lexicon of table 4 (with typed constant $\mathbf{j} : e$, $\mathbf{p} : e$, $\mathbf{in} : e \multimap (t \multimap t)$ and $\mathbf{live} : e \multimap t$). We want to parse the following sentence: *John lives in Paris*. We succeed in parsing this sentence if we can find axiom links making a correct proof net with lexical entries as input (negative conclusions) and S as output (positive conclusion). Here, correct means that there is no alternate elementary cycle nor crossing between axiom links. Figure 3 shows the correctness of this sentence.

Then we can deduce the semantic proof net before cut-elimination (figure 4(a)), and the semantic proof net after cut-elimination (figure 4(b)). The corresponding λ-term is as expected (**in p**)(**live j**). This example strictly follows [dG et al.96].

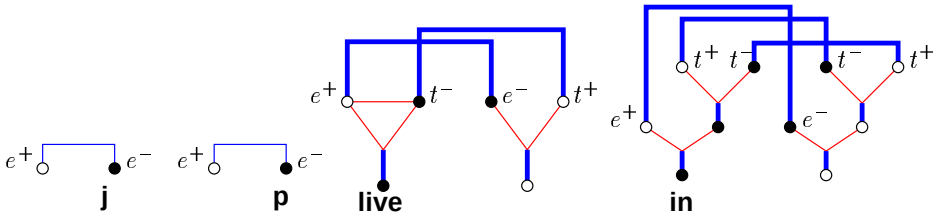
Table 4: Lexicon

lexical entry	syntaxic category	associated λ-term	semantic proof net
John	*NP*	**j**	Π_{John} (cf. figure 2(a))
Paris	*NP*	**p**	Π_{Paris} (cf. figure 2(b))
Lives	$NP\backslash S$	$\lambda x.\mathbf{live}\ x$	Π_{live} (cf. figure 2(c))
In	$(S\backslash S)/N$	$\lambda x.\lambda y.(\mathbf{in}\ x)y$	Π_{in} (cf. figure 2(d))

2.2 Generation

Let us now consider the problem of generation. We have a given semantic proof net (like the one in figure 4(b)) and we want to gather syntactic entries with axiom links such that:

1. this leads to a correct (syntactic) proof net;



(a) Π_{John} (b) Π_{Paris} (c) Π_{live} (d) Π_{in}

Figure 2: Semantic proof nets of the lexicon of table 4

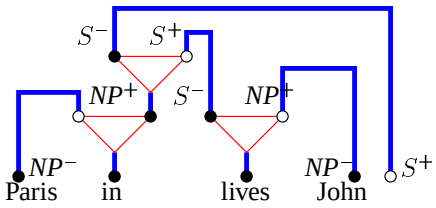
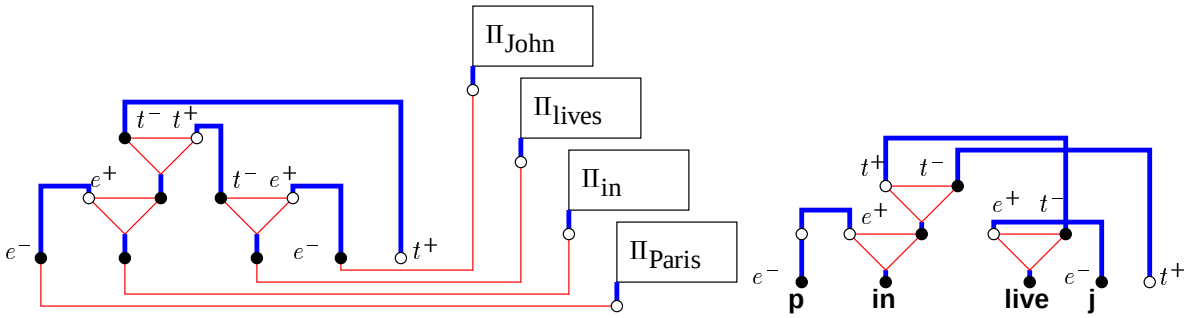


Figure 3: Syntactic proof net for *John lives in Paris*



(a) Before cut-elimination (b) After cut-elimination

Figure 4: Semantic proof nets for (**in p**)(**live j**)

2. the meaning of the resulting proof net matches the given semantic expression.

As we already said it, we assume that we have some lexical entries, and we try to make the generation with these entries, each one used once and only once.

Thus, if we define:

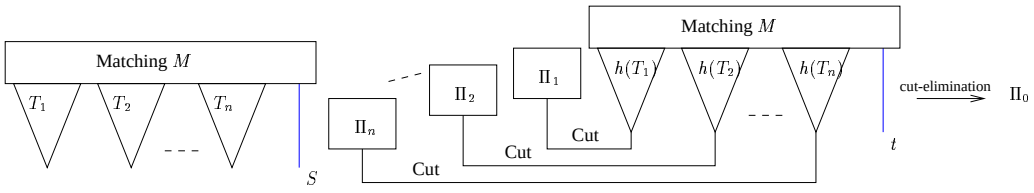
- Π_0 the semantic proof net of the expression we want to generate;
- π_i the semantic proof nets associated to the given lexical entries i we use;
- T_i the unfolding in proof structure of the syntactic formula of the lexical item i;
- F the forest made of the syntactic trees of all the considered lexical entries plus the output (the type we want to derive).

The generation problem (see figure 5) is to find a matching M of atomic formulas of F such that:

1. F endowed with M (let us call this proof structure F') is a correct proof net;
2. when cut-linking $H(F')$ with the Π_i, and eliminating these cuts, we obtain Π_0.

This problem is not an original one: making proof search with proof nets always leads to look for matching between atomic formulas of opposite polarities. So that a brutal answer to this problem would consist in taking F and try every possible matching. This brutal technic would of course appear essentially inefficient, and our purpose is to use everything we know to prune the search domain.

Nevertheless, note that even with such a brute force algorithm, we already reach the decidability (because the finitness of the number of the matchings) *without* making any assumption on the form of the semantic entries (neither on the order of the associated λ-terms, nor the linearity of these latter, nor the presence of a free variable). And we want to keep these good properties in our algorithm.



(a) A (syntactically) correct proof net

(b) The semantic relation

Figure 5: The generation problem

3 Cut-elimination as Matrix Computation

In this section, we consider the generation problem as a proof search problem. Working in an algebraic framework, we state the problem as an equation resolution. Then two cases appear, only depending on the form of the semantic proof nets of the given lexical entries.

In the first case, we show the solution, when it exists, is unique and we can compute it with a simple matrix product. In the second case, many solutions are possible and the complexity to obtain them is higher.

Note: We call $\mathcal{M}_{n,p}(\mathbb{K})$ the set of the matrices with n rows and p columns with their values in $\mathbb{K}$, and $\mathcal{M}_n(\mathbb{K}) = \mathcal{M}_{n,n}(\mathbb{K})$

3.1 Principles

This section presents a graphical, and then a matricial approach of cut-elimination on proof nets. It recalls the notions expressed initially in [Girard89] and reformulated in [Retore90, Girard93, Girard95].

Let us assume we have a proof net endowed with cut links. The cut elimination rewriting rules are twofolds (see section 1.1): the first identifies axiom links (when the cut directly holds between two axiom links), and the other enables the propagation of a cut on two dual formulas to the atoms that belong to these formulas. When performing recursively this latter rule, all the cuts between non atomic formulas disappear, and it only remains cut links between atoms. Since this process is confluent and does not depend on the axiom linking, we can consider it has no influence on the proof search process we are dealing with, hence we shall now consider that on the proof nets we are speaking about, only cut links between axiom links remain.

Let us consider such a proof net $\overline{U}$. We denote by $(e_i)_{1 \leq i \leq s}$ all the vertices taking place for atoms in $\overline{U}$. We can define U the incidence matrix of *axiom links*, σ the incidence matrix of *cut links* (only between axiom links, wrt. our previous comment), and Π the incidence matrix of axiom links of $\overline{\Pi}$ where $\overline{\Pi}$ is the proof net resulting from all the cut

eliminations on $\overline{U}$: $\forall (i,j) \in [1,s]^2, u_{ij} = 1$ (resp. $\pi_{ij} = 1$) iff there is an axiom link between e_i and e_j in $\overline{U}$ (resp. in $\overline{\Pi}$), and 0 elsewhere. It defines $U \in \mathcal{M}_s(\mathbb{R})$ (resp. Π).

And for $\sigma \in \mathcal{M}_s(\mathbb{R})$: $\forall (i,j) \in [1,s]^2, \sigma_{ij} = 1$ iff there is a cut link between e_i and e_j in $\overline{U}$, and 0 elsewhere.

Note that U, Π and σ are symmetric ($^tU = U, {}^t\Pi = \Pi. {}^t\sigma = \sigma$). Moreover, U and Π are involutions ($U^2 = 1, \Pi^2 = 1$): they express symmetric graph relations.

In a classical interpretation of incidence matrices, for all k, the matrix $U(\sigma U)^k$ is the matrix of the graph relation expressing that two vertices are linked if and only if when merging U and σ there exists a path starting and ending on an edge of U, and taking alternatively an edge in U and an edge in σ (k times).

If the proof net is correct, there cannot be any cycle (i.e. all such paths are finite). So there exists n_0 such that $\forall n \geq n_0, (\sigma U)^n = 0$ and σU is nilpotent.

Now, $\text{Ex}(U,\sigma) = U + U\sum_1^\infty (\sigma U)^k$ is the matrix for the graph expressing the relation that between to atoms, there exists a path of the required form (i.e. starting and ending with an edge of U, and taking alernatively an edge in U and an edge in σ), whatever its length is (possibly 1). And if both the starting and the ending vertice are not premises of a cut link, this corresponds exactly to the fact that after the cut-elimination process, these two atoms are linked by an axiom link.

So, to obtain exaclty the axiom links after cut-elimination, we have to suppress all the links coming from and arriving on an atom which is premise of a cut link. In other words, we consider the paths that *start* and *end* on vertices not involved in σ. Since σ^2 is the identity on those atoms, and zero elsewhere, $(1-\sigma^2)$ is zero on atoms involved in cut links, and the identity elsewhere. So that we have to multiply $\text{Ex}(U,\sigma)$ on the right and on the left by $(1-\sigma^2)$.

Then, if $\text{Res}(U,\sigma)$ is the matrix expressing the axiom links of the proof net obtained from $\overline{U}$ after cut-elimination (i.e. $\overline{\Pi}$), we have

$$\text{Res}(U,\sigma) = (1-\sigma^2)U(1+\sum_1^\infty (\sigma U)^k)(1-\sigma^2) \tag{1}$$

Moreover, since (σU) is nilpotent, $(1+\sum_1^\infty (\sigma U)^k)$ in inversible, and its inverse is $(1-\sigma U)$. The next section make explicit the relation (1) with a special choise of the base (e_i).

3.2 Matricial Relation for Cut Elimination

In the problem we are dealing with, we know $\overline{\Pi}$ and some of the axiom links in $\overline{U}$. Let us assume that $\forall i \in [1,p]$, both e_i and $B(e_i)$ [1] are not cut-linked in $\overline{U}$. Then we have (with $U_0 \in \mathcal{M}_p(\mathbb{R})$):

$$U = \left[\begin{array}{c:c} U_0 & 0 \\ \hdashline 0 & U_1 \end{array}\right] \text{ and } \sigma = \left[\begin{array}{c:c} 0 & 0 \\ \hdashline 0 & \sigma_1 \end{array}\right]$$

And $\text{Res}(\sigma,U) = \left[\begin{array}{c:c} U_0 & 0 \\ \hdashline 0 & \text{Res}(U_1,\sigma_1) \end{array}\right] = \Pi$. So we can consider without loss of generality that every axiom link of $\overline{U}$ has at least one of its conclusions in a cut link (atoms from axiom links not involved in cut elimination do not change from $\overline{U}$ to $\overline{\Pi}$, hence provide no valuable information).

3.3 Expressing the Reduction of U into Π

3.3.1 Some Preliminary Results

Proofs of these results are in appendix A.

Lemma 1 *let* $X \in \mathcal{M}_s(\mathbb{R})$ *such that* $X = \left[\begin{array}{c:c:c} 0 & 0 & 0 \\ \hdashline a & 0 & c \\ \hdashline b & 0 & d \end{array}\right]$ *then for all* $k \geq 2$

$$X^k = \left[\begin{array}{c:c:c} 0 & 0 & 0 \\ \hdashline cd^{k-2}b & 0 & cd^{k-1} \\ \hdashline d^{k-1}b & 0 & d^k \end{array}\right]$$

[1] $B(e)$ is the atom in $\overline{U}$ such that there is an axiom link between e and $B(e)$.

Corollary 1 *X (as above) is nilpotent (and then $(1-X)$ is inversible) if and only if d is nilpotent (and then $(1-d)$ is inversible). In this case*

$$(1-X)^{-1} = \left[\begin{array}{c:c:c} 1 & 0 & 0 \\ \hdashline a+c(1-d)^{-1}b & 1 & c(1-d)^{-1} \\ \hdashline (1-d)^{-1}b & 0 & (1-d)^{-1} \end{array}\right]$$

3.3.2 Reformulating the Relation

In this section, we want to give an relation equivalent to (1) which focuses on some axiom links we are interested in.

As mentioned in section 3.2, we can consider the (e_i) such that in $\overline{U}$:

- $\forall i \in [1,p]$, e_i is not cut-linked (then, because of the hypothesis made in section 3.2, $B(e_i)$ is cut-linked);
- $\forall i \in [p+1, p+m]$, e_i is cut-linked but $B(e_i)$ is not cut-linked;
- $\forall i \in [p+m+1, p+m+n]$, both e_i and $B(e_i)$ are cut-linked.

Note: Remember we assume that there is no axiom link such that both its conclusions are not cut-linked. So $p = m$.

Then in this base, we express the matrices (every axiom link of $\overline{U}$ has at least one of its conclusion involved in a cut link):

$$U = \left[\begin{array}{c:c:c} 0 & U_1 & 0 \\ \hdashline {}^tU_1 & 0 & 0 \\ \hdashline 0 & 0 & U_3 \end{array}\right] \quad \sigma = \left[\begin{array}{c:c:c} 0 & 0 & 0 \\ \hdashline 0 & \sigma_1 & \sigma_2 \\ \hdashline 0 & \sigma_3 & \sigma_4 \end{array}\right]$$

$$\Pi = \left[\begin{array}{c:c:c} \Pi_1 & 0 & 0 \\ \hdashline 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 \end{array}\right]$$

Of course, as $p = m$, we actually have that $U_1\,{}^tU_1 = {}^tU_1U_1 = 1$ (we use this property further). Note as well that if $\sigma_4 = 0$ then $n \leq m$. Moreover, by definition there is exactly a 1 per row and per column after the $p+1$th row and column in σ. So if $\sigma_4 = 0$ then every 1 is in σ_2 and the rank of σ_2 is n.

We then have $\sigma U = \left[\begin{array}{c:c:c} 0 & 0 & 0 \\ \hdashline \sigma_1\,{}^tU_1 & 0 & \sigma_2U_3 \\ \hdashline \sigma_3\,{}^tU_1 & 0 & \sigma_4U_3 \end{array}\right]$ and the definition of σ ensures that

$$\sigma^2 = \left[\begin{array}{c:c:c} 0 & 0 & 0 \\ \hdashline 0 & 1 & 0 \\ \hdashline 0 & 0 & 1 \end{array}\right] \text{ and } 1-\sigma^2 = \left[\begin{array}{c:c:c} 1 & 0 & 0 \\ \hdashline 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 \end{array}\right]$$

Lemma 2

$$\mathit{Res}(\sigma, U) = (1-\sigma^2)U(1-\sigma U)^{-1}(1-\sigma^2)$$

$$\Leftrightarrow$$

$$({}^tU_1\Pi_1 - \sigma_1\,{}^tU_1)U_1 = \sigma_2U_3(1+\sum_{k\geq 1}(\sigma_4U_3)^k)\,{}^t\sigma_2 \tag{2}$$

Let us define $A = ({}^tU_1\Pi_1 - \sigma_1\,{}^tU_1)U_1$ and $X = (U_3 - \sigma_4)^{-1}$. A is symmetric, X is symmetric and inversible and $U_3 = X^{-1} + \sigma_4$ and

$$\begin{aligned} U_3X &= 1 + \sigma_4X \\ &= 1 + (\sigma_4U_3)(U_3X) \\ &= 1 + \sum_{k\geq 1}(\sigma_4U_3)^k \end{aligned}$$

With the previous definitions, we then can state the theorem:

Theorem 1 *Let $\overline{U}$ be a correct proof net reducing in Res(σ, U) after cut elimination. These relations are equivalent:*

- *Res*$(\sigma, U) = (1-\sigma^2)U(1-\sigma U)^{-1}(1-\sigma^2)$
- $({}^tU_1\Pi_1 - \sigma_1\,{}^tU_1)U_1 = \sigma_2 U_3(1+\sum_{k\geq 1}(\sigma_4 U_3)^k)\,{}^t\sigma_2$
- $A = {}^t\sigma_2 X \sigma_2$ *and* $U_3 = X^{-1} + \sigma_4$.

Of course, all the terms are defined.

We base the proof search algorithm corresponding to the generation process we are dealing with on this third relation, as explained in the next sections.

Note: If $\sigma_4 = 0$, we have $U_3 = X$ (indeed, in this case, $U_3 = X^{-1}$. But $U_3^2 = 1$, so that $U_3^{-1} = U_3$).

3.4 Solving the Equations

In this section (proof search oriented), we consider that the axiom links we are looking for are those whose both conclusions are involved in cut links. That is we want to complete U_3. We first express an equivalent relation that allow us to characterize two different cases: the existence of one single solution and the existence of a set of solutions.

As in the previous section we proceeded by equivalence, solving the equation (2) in U_3 correponds to solving the equation

$$A = \sigma_2 X\,{}^t\sigma_2 \tag{3}$$

in X with X inversible. Then, we have to solve

$$U_3 = X^{-1} + \sigma_4$$

such that ${}^tU_3 = U_3$ and $U_3^2 = 1$.

From the equation (3) where we know A, the difficulties come from $\sigma_2 \in \mathcal{M}_{m,n}(\mathbb{R})$ which is not generally a square matrix nor inversible.

On the other hand it exists $P \in \mathcal{M}_m(\mathbb{R})$ and $Q \in \mathcal{M}_n(\mathbb{R})$ such that $\sigma_2 = PI_rQ$ ($r \leq \min(m,n)$ is the rank of σ_2 and $I_r \in \mathcal{M}_{m,n}$ is such that $\forall i \in [1,r] I_{rii} = 1$ and 0 elsewhere). Then with $B = P^{-1}A\,{}^tP^{-1}$ and $Y = QX\,{}^tQ$, the relation (3) is equivalent to

$$B = I_r Y\,{}^t I_r \tag{4}$$

where B is known. And from Y, we have $X = Q^{-1}Y\,{}^tQ^{-1}$ straightforwardly with Q we know.

Because I_r is not square, there are generally many solutions for Y. In particular, since $B \in \mathcal{M}_m(\mathbb{R})$ and $Y \in \mathcal{M}_n(\mathbb{R})$ we can not recover all the n^2 parameters of Y from the m^2 of B when $n > m$.

But what happens when $n \leq m$? If $r < n$ we have the same kind of problem because the relation (4) underspecifies Y. We shall not developp on these two cases, because we think it requires a specific work to adress these problems efficiently.

Nevertheless the case were $\sigma_4 = 0$, as previously noted, is such that $r = n \leq m$. And then

$$I_r Y\,{}^t I_r = \left[\begin{array}{c:c} Y & 0 \\ \hdashline 0 & 0 \end{array}\right] = B$$

which completely determines Y and then X.

Moreover, $\sigma_4 = 0$ is equivalent to the fact that in $\overline{U}$ no axiom link whose both conclusions are involved in cut links is cut-linked to an axiom of the same kind. But where such a configuration could occur ? Since cuts are *only* between semantic proof nets and the image of the syntactic forest, it would necessitate that in at least a semantic proof net, an axiom link has both its conclusions involve in a cut, that is whose conclusions appear in the unique output of the semantic proof net.

So depending on the given lexical items, we know whether we will have to consider a unique solution polyniomally obtained, or if we have to consider a set of solutions.

4 Example

Let us process on an example the previous results. We still use the lexicon of table 4 on page 6, and we want to generate (if possible) a sentence whose meaning is given by the proof net of figure 4 on page 7.

We first need to associate every atom with an index (in the figures, we only indicate a number i beside the atom to express it is e_i). Of course, we have to know how to recognize the e_i that are the same in U (figure 6(b) on the facing page) and in Π (figure 6(a) on the next page). This can be done by looking at the typed constants decorating the input conclusions[2].

We also assume in this numerotation that we know which of the atoms in $H(F)$ is linked to t^+ (the unique output) i.e. we assume we know which of the atoms of $H(F)$ is in U_1. We don't have a clever way to find it for the moment, but in the worse case (if we try every possibility), the polynomiality (in the number of atoms) of the procedure increases by one.

Then, the given matrices are:

$$U_1 = \begin{bmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&1 \end{bmatrix} \qquad \Pi_1 = \begin{bmatrix} 0&1&0&0&0&0&0&0\\ 1&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&0&1\\ 0&0&1&0&0&0&0&0\\ 0&0&0&0&0&1&0&0 \end{bmatrix}$$

$$\sigma_1 = \begin{bmatrix} 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0 \end{bmatrix} \qquad \sigma_2 = \begin{bmatrix} 1&0&0&0&0&0\\ 0&1&0&0&0&0\\ 0&0&1&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&1&0\\ 0&0&0&1&0&0\\ 0&0&0&0&0&1 \end{bmatrix} = \begin{bmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&1\\ 0&0&0&0&0&1&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0 \end{bmatrix} I_6 \begin{bmatrix} 1&0&0&0&0&0\\ 0&1&0&0&0&0\\ 0&0&1&0&0&0\\ 0&0&0&1&0&0\\ 0&0&0&0&0&1\\ 0&0&0&0&1&0 \end{bmatrix}$$

So

$$A = ({}^tU_1\Pi_1 - \sigma_1\,{}^tU_1)U_1 = \begin{bmatrix} 0&1&0&0&0&0&0&0\\ 1&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&1\\ 0&0&1&0&0&0&0&0\\ 0&0&0&0&0&1&0&0 \end{bmatrix}$$

and

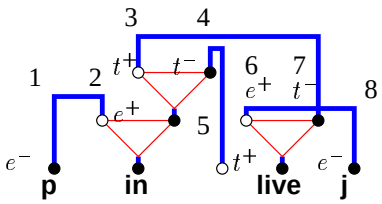
$$B = P^{-1}A\,{}^tP^{-1} = \begin{bmatrix} 0&1&0&0&0&0&0&0\\ 1&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&1&0&0&1&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0 \end{bmatrix}$$

So

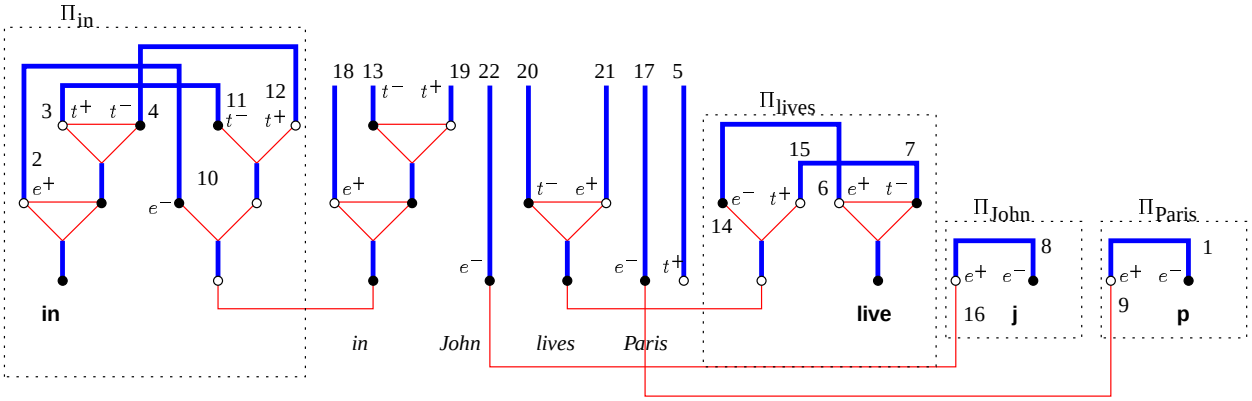
$$X = Q^{-1}Y\,{}^tQ^{-1} = U_3 = \begin{bmatrix} 0&1&0&0&0&0\\ 1&0&0&0&0&0\\ 0&0&0&1&0&0\\ 0&0&1&0&0&0\\ 0&0&0&0&0&1\\ 0&0&0&0&1&0 \end{bmatrix}$$

[2] This is of course a necessary step that we have not discuss so far. And we plan to deal with this problem in further work.

We can add this matching to the syntactic forest of figure 7(a) on this page (do not forget that the link between S^+ and S^- is in U_1 and not in U_3, and that U_3 represents edges between e_i with $i \in [17, 22]$) and obtain on F the matching of figure 7(b) on the current page.

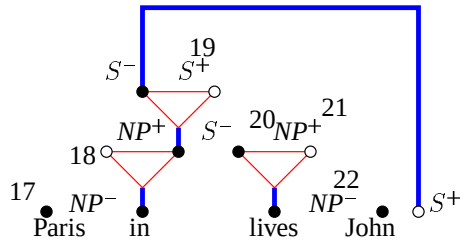


(a) Marking atoms on Π

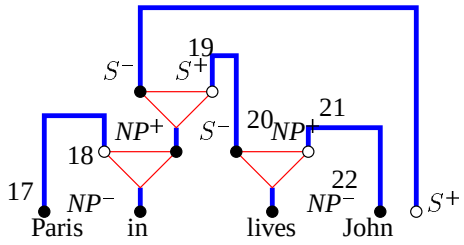


(b) Marking atoms on U

Figure 6: Defining the base (e_i) on $\overline{U}$ and $\overline{\Pi}$



(a) The syntactic forest



(b) The syntactic forest endowed with the matching described by U_3

Figure 7: Applying the matching on the syntactic forest

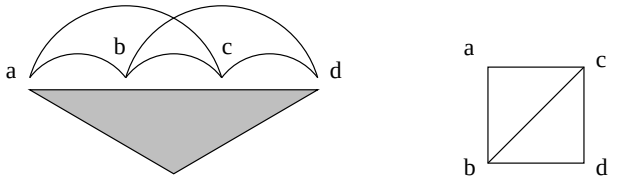
We still have to ensure the correctness of this proof net (because we add all the tensor and par links), but it has a quadratic complexity (less than the matricial computation). In this case, it is correct. Moreover, U_3 proposes links between dual atoms.

Note: Actually, this only gives us the axiom links. It still necessitates to compute the word order (to check if there exists a configuration in which axiom links do not cross one another, because we shall not always have the fortune to numerotate them in the right order). This is the subject of the next section.

5 Managing Word Order

In the previous sections, we saw the way of computing axiom links between atoms of the syntactic forest, and we know if this leads to a correct *commutative* proof net. Of course, in the case of Lambek calculus, this is not enough: we have to check wether it exists an order of the lexical item that makes this proof net without any crossing axiom link. This section describes how to find such an order if it exists.

In fact, we explained in the proof net for Lambek calculus presentation that, on top of the abscence of any ae-cycle, the correction requires a planarity criterion. But this planarity does not exactly match the usual planarity concept in graph theory because in our case we have stronger constraints: vertices from the same word must remain aligned in the same order and edges in the same semi-plan. For example considering the graph of axiom links (between a, b, c and d) on figure 8(a) (which we can assume to belong to the proof structure represented by the grey triangle; the little black bows denote the fixed order within a word and the big ones denote axiom links) this graph should be considered as planar since it admits a representation (see figure 8(b)) in the plan with no crossing edge. But of course, we actually want to reject such a proof structure. So we can not strictly apply classical and linear planarity testing algorithms.



(a) Proof structure representation

(b) Planar representation

Figure 8: Planar drawing

This divides the word order search in two parts: within words (which is actually more a check since this order is given by the syntactic category: we need to check the proposed matching compatibility) and between words. Succeeding in word order search means we can express this semantical expression correctly with the lexical choice, and failing means we can't express the given semantic expression with this lexical choice.

Problems within words and between words being slightly different, the two next sections deal respectively with each one of them to present their respective part in the search order algorithm.

5.1 Correction within Words

This part of the algorithm is not exactly devoted to word order computation, for axiom links within words does not occur in placing words relatively one to another. But it provides both a first filter to reject some matchings and a preprocess for the actual word order treatment. This preprocess consists in droping all the axiom links between atoms of a same lexical item so that only relevant axiom links for the word order remain.

Notation: The unfolding of the syntactic category of a lexical item gives a tree whose leaves are atomic formulas. In the following, we generally speak about the *word* made from the sequence of these atoms.

Definition 1 *By analogy with free groups, we say that w is* reducible *if;*

- $w = aB(a)$ *($B : x \to B(x)$ such that both x and $B(x)$ are the conclusions of the same axiom link);*

- $w = aw'B(a)$ *and* w' *is reducible;*
- $w = w_1 w_2$ *and* w_1 *and* w_2 *are reducible.*

For a single word $w = a_1 \dots a_n$, we can formulate the planarity property for lambek calculus as a property on w: $w = a_1 \dots a_i w' a_j \dots a_n$ (with the convention that if $i = 0$ then $w = w' a_j \dots a_n$ and if $j = m+1$ the $w = a_1 \dots a_i w'$) and $\forall k \in [1, i] \cup [j, n]$, $B(a_k)$ is not in w and w' is reducible. This simply translates the correct bracketting property of atoms expressed in [Lamarche et al.96].

And $a_1 \dots a_i a_j \dots a_n$ is the *reduced* form of w, the one that will actually play a role in the word order computation. By extension, we also say that w is reducible.

So the preprocessing part consists in checking that every word is reducible and in computing its reduced form. Recovering the second atom of an axiom link can be done in $(\log mn)$ where m is the number of selected lexical entries and n the greatest valence of the selected lexical entries so that this modification to usual bracketting algorithms makes them run in $(mn \log mn)$ for all the word preprocessings.

If this first part of the algorithm succeed, we then only consider words *without* their reducible part (reduced words). It means that only what plays a role in the word ordering remains

5.2 Correction between Words

To deal with the word ordering, we use two main properties of a correct matching: we can arbitrarily decide what is the rightmost word (so we take the one corresponding to the formulas we want to prove, usually S).

The second one is a little bit more complex (see figure 9): given a word $w = a_1 \dots a_n$ such that a_i is linked with an axiom link to the rightmost word w_0 then if w_k is the word linked to a_k, $\forall k \in [1, i-1]$ w_k is on the left of w (which becomes the rightmost word for these words) and $\forall k \in [i+1, n]$ w_k is on the right of w (which becomes the leftmost word of these words, with always the same rightmost word).

Then again, in their delimited domain, w_k are rightmost or leftmosts words and we can apply the same property.

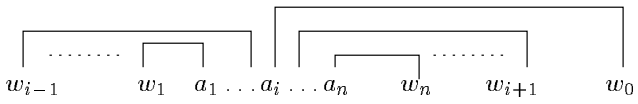


Figure 9: How axiom links induce word order

The algorithm 1 then checks if an order exists (to make the proof net correct in the non commutative case) and builds it if possible. The start is Order$(w_0, \text{length}(w_0), \text{Left}, G, \text{null}, \text{null})$ with G the set of all the words minus w_0 and we get a list ordered as should be the lexical items from left to right in a $(mn \log mn)$ time.

(In this algorithm, word(x) is a function that computes the word to which $B(x)$ belongs and place(x) computes the position of $B(x)$ in word(x).)

Algorithm 1 Order(b, i, orientation, S, a, j_0)

$L' \leftarrow \emptyset$
$L'' \leftarrow \emptyset$
$m =$ word($b[i]$)
$j =$ place(lien($b[i]$))
if $m \notin S$ **then** {if the word was already placed}
  if $m \neq a$ **then** {and it is not the last pivot}
    FALSE
  else if orientation=Left **then**
    if $j - 1 \neq j_0$ **then** {there is a gap between links between a and b}
      FALSE
    else if $i > 1$ **then** {try with the next atom on the left of b}
      order($b, i - 1$, orientation, S, a, j)
    else if $j <$ length(a) **then** {try with the next atom on the right of a}
      order($a, j + 1$, Right, S, null, null)
    end if
  else if $j + 1 \neq j_0$ **then** {there is a gap between links between a and b}
    FALSE
  else if $i <$ length(b) **then** {try with the next atom on the right of b}
    order($b, i + 1$, orientation, S, a, j)
  else if $j > 1$ **then** {try with the next atom on the left of a}
    order($a, j - 1$, Left, S, null, null)
  end if
else
  if orientation $=$ Left **then**
    pivotL$\leftarrow (a, j_0)$; pivotR$\leftarrow (b, i)$
  else
    pivotR$\leftarrow (a, j_0)$; pivotL$\leftarrow (b, i)$
  end if
  if $j > 1$ **then** {order on the left of m}
    $L' \leftarrow$ Order($m, j - 1$, Left, $S \backslash \{m\}$, pivotL)
  end if
  if $j <$ length(m) **then** {order on the right of m}
    $L'' \leftarrow$ Order($m, j - 1$, Left, $S \backslash \{m\}$, pivotR)
  end if
end if
Return $L' + [m] + L''$

Conclusion

We took advantage of proof nets on the semantic point of view and we expressed the generation process as a guided proof search. On top of keeping the decidability property of this framework, we characterized the semantic proof nets that enable a polynomial time process.

Such properties are crucial because it is the central part of the generation process (considering Lambek calculus). But there are other things left to look at. As the very next steps, we should work on the atoms numbering and the choice of the lexical items. Appropriate interactions between word order contraints and matricial resolution in the hard case should also be considered. Moreover, another point is to benefit from the power of linear logic and deal with non linear λ-terms.

Finally, since some works [Moortgat96, Lecomte et al.95] consider different extensions of Lambek calculus based on proof nets, we hope our proposal and its good properties to apply to other linguistic approaches.

Acknowledgments

I would like to thank Christian Retoré who pointed out to me the Girard's algebraic interpretation of the cut elimintation.

References

[Ajdukiewicz35] K. Ajdukiewicz. – Die syntaktische Konnexitat. *Studia Philosophica*, vol. 1, 1935, pp. 1–27. – English translation in Storrs McCall (ed), *Polish Logic 1920-1939*, Oxford University Press, pp. 207-231.

[Danos90] V. Danos. – *Une Application de la Logique Linéaire à l'Étude des Processus de Normalisation (principalement du λ-calcul)*. – PhD thesis, Université Paris VII, June 1990.

[dG et al.96] P. de Groote et C. Retoré. – On the semantic readings of proof-nets. *In : Formal Grammar*, éd. par G. M. Geert-Jan Kruijff et D. Oehrle. FoLLI, pp. 57–70. – Prague, August 1996.

[Dowty et al.81] D. R. Dowty, R. E. Wall et S. Peters. – *Introduction to Montague Semantics*. – Kluwer Academic Publishers, 1981.

[Girard et al.88] J.-Y. Girard, Y. Lafont et P. Taylor. – *Proofs and Types*. – Cambridge University Press, 1988, *Cambridge Tracts in Theoretical Computer Science 7*.

[Girard87] J.-Y. Girard. – Linear logic. *Theoretical Computer Science*, vol. 50, 1987, pp. 1–102.

[Girard89] J.-Y. Girard. – Geometry of interaction I: Interpretation of system F. *In : Logic Colloquium '88*, éd. par C. Bonotto, R. Ferro, S. Valentini et A. Zanardo. pp. 221–260. – North-Holland.

[Girard93] J.-Y. Girard. – Linear logic: Its syntax and semantics. *In : Advances in Linear Logic*, éd. par J.-Y. G. N. Y. Lafont et L. Regnier. – Ithaca, New York, June 1993.

[Girard95] J.-Y. Girard. – Geometry of interaction III: The general case. *In : Advances in Linear Logic*, éd. par J.-Y. Girard, Y. Lafont et L. Regnier, pp. 329–389. – Cambridge University Press, 1995. Proceedings of the Workshop on Linear Logic, Ithaca, New York, June 1993.

[Howard80] W. A. Howard. – *To H. B. Curry: Essays on combinatory logic, Lambda Calculus and Formalism*, chap. The Formulæ-as-Types Notion of Construction, pp. 479–490. – Academic Press, 1980.

[Lamarche et al.96] F. Lamarche et C. Retoré. – Proof-nets for the lambek calculus – an overview. *In : Proceedings 1996 Roma Workshop. Proofs and Linguistic Categories*, éd. par V. M. Abrusci et C. Casadio. pp. 241–262. – Editrice CLUEB, Bologna.

[Lamarche95] F. Lamarche. – Games semantics for full propositional linear logic. *In : Proceedings, Tenth Annual IEEE Symposium on Logic in Computer Science*. IEEE Computer Society Press, pp. 464–473. – San Diego, California, 26–29 June 1995.

[Lambek58] J. Lambek. – The mathematics of sentence structure. *American Mathematical Monthly*, vol. 65 (3), 1958, pp. 154–170.

[Lecomte et al.95] A. Lecomte et C. Retoré. – Pomset logic as an alternative categorial grammar. *In : Formal Grammar*. – Barcelona, 1995.

[Merenciano et al.97] J. M. Merenciano et G. Morrill. – Generation as deduction on labelled proof nets. In Retoré [Retore97], pp. 310–328.

[Montague74] R. Montague. – *Formal Philosophy: Selected Papers of Richard Montague*. – New Haven, CT, Yale University Press, 1974.

[Moortgat96] M. Moortgat. – Categorial type logics. *In : Handbook of Logic and Language*, éd. par J. van Benthem et A. ter Meulen, pp. 5–91. – Amsterdam, Elsevier Science Publishers, 1996.

[Retore90] C. Retoré. – A note on turbo cut elimination. – September 1990. Manuscript.

[Retore93] C. Retoré. – *Réseaux et séquents ordonnés*. – PhD thesis, University of Paris VII, 1993.

[Retore97] C. Retoré, editor. – *Logical Aspects of Computational Linguistics: First International Conference, LACL '96, Nancy, France, September 23–25, 1996: selected papers*. – New York, NY, USA, Springer-Verlag Inc., 1997, viii + 434p.

[Roorda91] D. Roorda. – *Resource Logics: Proof-theoretical Investigations*. – PhD thesis, University of Amsterdam, September 1991.

A Few Results on Matricial Computation

Notation: In the following, to express that $X \in \mathcal{M}_{u,v}(\mathbb{R})$, we generally note $X^{(u,v)}$.

Lemma 3 *Let $X \in \mathcal{M}_s(\mathbb{R})$. If X is nilpotent ($\exists n_0, X^{n_0} = 0$) then $(1-X)$ is inversible and $(1-X)^{-1} = 1 + \sum_{k\geq 1} X^k$*

Proof: $\sum_{k\geq 1} X^k = \sum_{k=1}^{n_0} X^k$ is defined and

$$\begin{aligned}(1-X)(1+\sum_{k\geq 1}) &= 1 + X + \sum_{k\geq 2}) - X - X\sum_{k\geq 1} \\ &= 1\end{aligned}$$

□

Lemma 4 (1) *let $X \in \mathcal{M}_s(\mathbb{R})$ such that $X = \begin{bmatrix} 0^{(p,p)} & 0^{(p,m)} & 0^{(p,n)} \\ a^{(m,p)} & 0^{(m,m)} & c^{(m,n)} \\ b^{(n,p)} & 0^{(n,m)} & d^{(n,n)} \end{bmatrix}$ then for all $k \geq 2$*

$$X^k = \begin{bmatrix} 0^{(p,p)} & 0^{(p,m)} & 0^{(p,n)} \\ cd^{k-2}b^{(m,p)} & 0^{(m,m)} & cd^{k-1(m,n)} \\ d^{k-1}b^{(n,p)} & 0^{(n,m)} & d^{k(n,n)} \end{bmatrix}$$

Proof: By induction on k :

$$\begin{aligned}X^2 &= \begin{bmatrix} 0^{(p,p)} & 0^{(p,m)} & 0^{(p,n)} \\ a^{(m,p)} & 0^{(m,m)} & c^{(m,n)} \\ b^{(n,p)} & 0^{(n,m)} & d^{(n,n)} \end{bmatrix} * \begin{bmatrix} 0^{(p,p)} & 0^{(p,m)} & 0^{(p,n)} \\ a^{(m,p)} & 0^{(m,m)} & c^{(m,n)} \\ b^{(n,p)} & 0^{(n,m)} & d^{(n,n)} \end{bmatrix} \\ &= \begin{bmatrix} 0^{(p,p)} & 0^{(p,m)} & 0^{(p,n)} \\ cb^{(m,p)} & 0^{(m,m)} & cd^{(m,n)} \\ db^{(n,p)} & 0^{(n,m)} & d^{2(n,n)} \end{bmatrix}\end{aligned}$$

Let us assume that $X^k = \begin{bmatrix} 0^{(p,p)} & 0^{(p,m)} & 0^{(p,n)} \\ cd^{k-2}b^{(m,p)} & 0^{(m,m)} & cd^{k-1(m,n)} \\ d^{k-1}b^{(n,p)} & 0^{(n,m)} & d^{n(n,n)} \end{bmatrix}$ then

$$\begin{aligned}X^{k+1} &= \begin{bmatrix} 0^{(p,p)} & 0^{(p,m)} & 0^{(p,n)} \\ a^{(m,p)} & 0^{(m,m)} & c^{(m,n)} \\ b^{(n,p)} & 0^{(n,m)} & d^{(n,n)} \end{bmatrix} * \begin{bmatrix} 0^{(p,p)} & 0^{(p,m)} & 0^{(p,n)} \\ cd^{k-2}b^{(m,p)} & 0^{(m,m)} & cd^{k-1(m,n)} \\ d^{k-1}b^{(n,p)} & 0^{(n,m)} & d^{k(n,n)} \end{bmatrix} \\ &= \begin{bmatrix} 0^{(p,p)} & 0^{(p,m)} & 0^{(p,n)} \\ cd^{k-1}b^{(m,p)} & 0^{(m,m)} & cd^{k(m,n)} \\ d^k b^{(n,p)} & 0^{(n,m)} & d^{k+1(n,n)} \end{bmatrix}\end{aligned}$$

□

Corollary 2 (1) *X (as above) is nilpotent (and then $(1-X)$ is inversible) if and only if d is nilpotent (and then $(1-d)$ is inversible). In this case*

$$(1-X)^{-1} = \begin{bmatrix} 1^{(p,p)} & 0^{(p,m)} & 0^{(p,n)} \\ a + c(1-d)^{-1}b^{(m,p)} & 1^{(m,m)} & c(1-d)^{-1(m,n)} \\ (1-d)^{-1}b^{(n,p)} & 0^{(n,m)} & (1-d)^{-1(n,n)} \end{bmatrix}$$

Proof: This is a trivial equivalence and if it is defined, $(1-X)^{-1}(1-X)=1$ □

Lemma 5 (2) *With U, σ and Π defined as in 3.2, we have*

$$\mathit{Res}(\sigma, U) = (1-\sigma^2)U(1-\sigma U)^{-1}(1-\sigma^2)$$

$$\Leftrightarrow$$

$$({}^tU_1\Pi_1 - \sigma_1\,{}^tU_1)U_1 = \sigma_2 U_3(1+\sum_{k\geq 1}(\sigma_4 U_3)^k)\,{}^t\sigma_2$$

Proof:

$$\mathrm{Res}(\sigma, U) = (1-\sigma^2)U(1-\sigma U)^{-1}(1-\sigma^2)$$

$$\Leftrightarrow$$

$$\Pi = \left[\begin{array}{c:c:c} 1^{(p,p)} & 0^{(p,m)} & 0^{(p,n)} \\ \hdashline 0^{(m,p)} & 0^{(m,m)} & 0^{(m,n)} \\ \hdashline 0^{(n,p)} & 0^{(n,m)} & 0^{(n,n)} \end{array}\right] * \left[\begin{array}{c:c:c} 0^{(p,p)} & {U_1}^{(p,m)} & 0^{(p,n)} \\ \hdashline {}^tU_1{}^{(m,p)} & 0^{(m,m)} & 0^{(m,n)} \\ \hdashline 0^{(n,p)} & 0^{(n,m)} & {U_3}^{(n,n)} \end{array}\right]$$

$$* \left[\begin{array}{c:c:c} 1^{(p,p)} & 0^{(p,m)} & 0^{(p,n)} \\ \hdashline a + c(1-d)^{-1}b^{(m,p)} & 1^{(m,m)} & c(1-d)^{-1\,(m,n)} \\ \hdashline (1-d)^{-1}b^{(n,p)} & 0^{(n,m)} & (1-d)^{-1\,(n,n)} \end{array}\right]$$

$$* \left[\begin{array}{c:c:c} 1^{(p,p)} & 0^{(p,m)} & 0^{(p,n)} \\ \hdashline 0^{(m,p)} & 0^{(m,m)} & 0^{(m,n)} \\ \hdashline 0^{(n,p)} & 0^{(n,m)} & 0^{(n,n)} \end{array}\right]$$

$$= \left[\begin{array}{c:c:c} 0^{(p,p)} & {U_1}^{(p,m)} & 0^{(p,n)} \\ \hdashline 0^{(m,p)} & 0^{(m,m)} & 0^{(m,n)} \\ \hdashline 0^{(n,p)} & 0^{(n,m)} & 0^{(n,n)} \end{array}\right]$$

$$* \left[\begin{array}{c:c:c} 1^{(p,p)} & 0^{(p,m)} & 0^{(p,n)} \\ \hdashline a + c(1-d)^{-1}b^{(p,p)} & 0^{(m,m)} & 0^{(m,n)} \\ \hdashline (1-d)^{-1}b^{(n,p)} & 0^{(n,m)} & 0^{(n,n)} \end{array}\right]$$

$$= \left[\begin{array}{c:c:c} U_1(a + c(1-d)^{-1}b)^{(p,p)} & 0^{(p,m)} & 0^{(p,n)} \\ \hdashline 0^{(m,p)} & 0^{(m,m)} & c0^{(m,n)} \\ \hdashline 0^{(n,p)} & 0^{(n,m)} & 0^{(n,n)} \end{array}\right]$$

With

$$\begin{aligned} a &= \sigma_1\,{}^tU_1 \\ b &= {}^t\sigma_2\,{}^tU_1 \\ c &= \sigma_2 U_3 \\ d &= \sigma_4 U_3 \end{aligned}$$

we have that

$$\Pi_1 = U_1(\sigma_1\,{}^tU_1 + \sigma_2 U_3(1+\sum_{k\geq 1}(\sigma_4 U_3)^k)\,{}^t\sigma_2\,{}^tU_1)$$

Since $U_1\,{}^tU_1 = 1 = {}^tU_1U_1$ we have

$$({}^tU_1\Pi_1 - \sigma_1\,{}^tU_1)U_1 = \sigma_2 U_3(1+\sum_{k\geq 1}(\sigma_4 U_3)^k)\,{}^t\sigma_2$$

□

Lemma 6 *If* $M = \left[\begin{array}{c|c|c} 0 & 0 & 0 \\ \hline 0 & 0 & a \\ \hline b & c & 0 \end{array}\right]$ *then*

$$\forall k \geq 0 \quad M^{2k+1} = \left[\begin{array}{c|c|c} 0 & 0 & 0 \\ \hline 0 & 0 & (ac)^k a \\ \hline (ca)^k b & (ca)^k c & 0 \end{array}\right]$$

and

$$\forall k \geq 1 \quad M^{2k} = \left[\begin{array}{c|c|c} 0 & 0 & 0 \\ \hline (ac)^{k-1} ab & (ac)^k & 0 \\ \hline 0 & 0 & (ca)^k \end{array}\right]$$

and

$$1 + \sum_1^\infty M^k = \left[\begin{array}{c|c|c} 1 & 0 & 0 \\ \hline (1-ac)^{-1} ab & (1-ac)^{-1} & (1-ac)^{-1} a \\ \hline (1-ca)^{-1} b & (1-ca)^{-1} c & (1-ca)^{-1} \end{array}\right]$$

Proof: $M^2 = \left[\begin{array}{c|c|c} 0 & 0 & 0 \\ \hline ab & ac & 0 \\ \hline 0 & 0 & ca \end{array}\right]$ $M^3 = \left[\begin{array}{c|c|c} 0 & 0 & 0 \\ \hline 0 & 0 & aca \\ \hline cab & cac & 0 \end{array}\right]$

$$M^{2k+2} = M^{2k+1} M = \left[\begin{array}{c|c|c} 0 & 0 & 0 \\ \hline 0 & 0 & (ac)^k a \\ \hline (ca)^k b & (ca)^k c & 0 \end{array}\right] \left[\begin{array}{c|c|c} 0 & 0 & 0 \\ \hline 0 & 0 & a \\ \hline b & c & 0 \end{array}\right] = \left[\begin{array}{c|c|c} 0 & 0 & 0 \\ \hline (ac)^k ab & (ac)^{k+1} & 0 \\ \hline 0 & 0 & (ca)^{k+1} \end{array}\right]$$

$$M^{2k+3} = M^{2k+2} M = \left[\begin{array}{c|c|c} 0 & 0 & 0 \\ \hline (ac)^k ab & (ac)^{k+1} & 0 \\ \hline 0 & 0 & (ca)^{k+1} \end{array}\right] \left[\begin{array}{c|c|c} 0 & 0 & 0 \\ \hline 0 & 0 & a \\ \hline b & c & 0 \end{array}\right] = \left[\begin{array}{c|c|c} 0 & 0 & 0 \\ \hline 0 & 0 & (ac)^{k+1} a \\ \hline (ca)^{k+1} b & (ca)^{k+1} c & 0 \end{array}\right]$$

We then have

$$
\begin{aligned}
1+\sum_1^\infty M^k &= 1+\sum_1^\infty \left[\begin{array}{c|c|c} 0 & 0 & 0 \\ \hline (ac)^{k-1}ab & (ac)^k & 0 \\ \hline 0 & 0 & (ca)^k \end{array}\right] + M + \sum_1^\infty \left[\begin{array}{c|c|c} 0 & 0 & 0 \\ \hline 0 & 0 & (ac)^k a \\ \hline (ca)^k b & (ca)^k c & 0 \end{array}\right] \\
&= 1+M+\sum_1^\infty \left[\begin{array}{c|c|c} 0 & 0 & 0 \\ \hline (ac)^{k-1}ab & (ac)^k & (ac)^k a \\ \hline (ca)^k b & (ca)^k c & (ca)^k \end{array}\right] \\
&= 1+M+\left[\begin{array}{c|c|c} 0 & 0 & 0 \\ \hline \sum_1^\infty (ac)^{k-1}ab & \sum_1^\infty (ac)^k & \sum_1^\infty (ac)^k a \\ \hline \sum_1^\infty (ca)^k b & \sum_1^\infty (ca)^k c & \sum_1^\infty (ca)^k \end{array}\right] \\
&= M+\left[\begin{array}{c|c|c} 1 & 0 & 0 \\ \hline \sum_0^\infty (ac)^k ab & 1+\sum_1^\infty (ac)^k & \sum_1^\infty (ac)^k a \\ \hline \sum_1^\infty (ca)^k b & \sum_1^\infty (ca)^k c & 1+\sum_1^\infty (ca)^k \end{array}\right] \\
&= \left[\begin{array}{c|c|c} 1 & 0 & 0 \\ \hline (1+\sum_1^\infty (ac)^k)ab & 1+\sum_1^\infty (ac)^k & (1+\sum_1^\infty (ac)^k)a \\ \hline (1+\sum_1^\infty (ca)^k)b & (1+\sum_1^\infty (ca)^k)c & 1+\sum_1^\infty (ca)^k \end{array}\right] \\
&= \left[\begin{array}{c|c|c} 1 & 0 & 0 \\ \hline (1-ac)^{-1}ab & (1-ac)^{-1} & (1-ac)^{-1}a \\ \hline (1-ca)^{-1}b & (1-ca)^{-1}c & (1-ca)^{-1} \end{array}\right]
\end{aligned}
$$

□

Contents

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399