Generation in the Lambek Calculus Framework: an Approach with Semantic Proof Nets

Sylvain Pogodalla
sylvain.pogodalla@xrce.xerox.com
Xerox Researce Centre Europe
6, chemin de Maupertuis
38240 Meylan — France

1 Introduction

The linear logic (Girard, 1987) provides a powerful framework to express categorial grammars (Ajdukiewicz, 1935) and Lambek calculus (Lambek, 1958), and a lot of work has presented proof nets uses for linguistic purposes, with a special look at proof nets for Lambek calculus (Roorda, 1991; Lamarche and Retoré, 1996). But they have mainly explored the syntactic capabilities of proof nets, describing parsing processes.

This paper wants to focus on the generation capabilities of proof nets thanks to their semantic readings as expressed in (de Groote and Retoré, 1996). The main features of our proposal consist in the use of proof nets for Lambek calculus, of the Curry-Howard isomorphism (Howard, 1980; Girard et al., 1988), of semantic proof nets with semantic expressions *à la* Montague (Montague, 1974; Dowty et al., 1981), and in an algorithm for proof search with a target proof net.

Unlike a previous proposal for generation in the Lambek calculus framework (Merenciano and Morrill, 1997), this point of view avoids the use of the λ-term unification to lead the generation process. And the algorithmic undecidability of this latter mechanism (from second order unification) does not occur any more.

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 the given semantic expression and generate a syntactically correct expression. For this purpose, we express our problem as a proof search one in (multiplicative) linear logic which is decidable. Moreover, we characterize the semantic recipes of lexical items that provide a polynomial solution for the generation process.

2 Multi Usage Proof Nets

2.1 Proof Nets

(Girard, 1987) introduced proof nets formalism as the natural deduction syntax for linear logic, also studied in (Retoré, 1993). 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 the rules introduction).

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 the leaves are formulas, e.g. atomic ones) and linking together the formulas occurring in the same axiom rule of the sequent calculus.

But proof nets have a more intrinsic definition that prevents 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. So that we do not present the sequent calculus but only the proof net calculus.

In this paper, we do not consider all the proof nets, but a part of the 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)* (Danos, 1990; Lamarche, 1995) 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 implications 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 that respect the correctness criterion.

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 the proof net formalism with a sim-

Table 1: Links

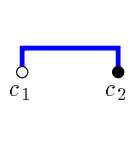
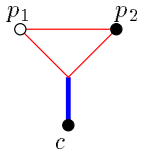
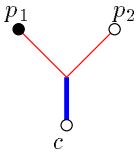
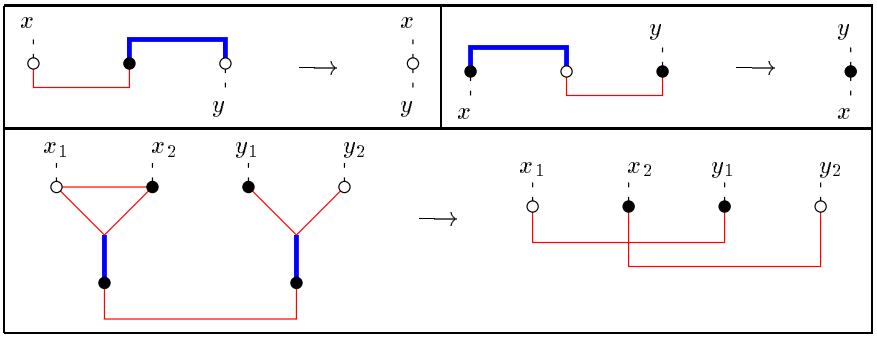
Name	*Axiom*	*Tensor*	*Par*	*Cut*
Link				
Premises	none	p_1, p_2	p_1, p_2	p_1, p_2
Conclusions	c_1, c_2	c	c	none
Types	$c_1 : A^+$ $c_2 : A^-$	$p_1 : A^+$ $p_2 : B^-$ $c : (A \multimap B)^-$	$p_1 : A^-$ $p_2 : B^+$ $c : (A \multimap B)^+$	$p_1 : A^-$ $p_2 : A^+$

Table 2: Cut-elimination rewriting rules



ple rewriting process described in table 2 (in case of complex formulas as in the third rewriting rule, those rules can apply again on the result and propagate until reaching atoms).

2.2 Syntactic Proof Nets

Definitions of proof nets for Lambek calculus first appeared in (Roorda, 1991). They naturally raised as Lambek calculus is an intuitionistic fragment of non commutative linar logic (with two linear implications: "$\backslash$" on the left and "$/$" on the right), and the consequences on the proof net calculus we presented in section 2.1 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.

If T_p is the set of basic types (e.g. $S, NP \ldots$), the set T of syntactic types follows $T ::= T_p | T\backslash T | T/T$.

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 non 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.

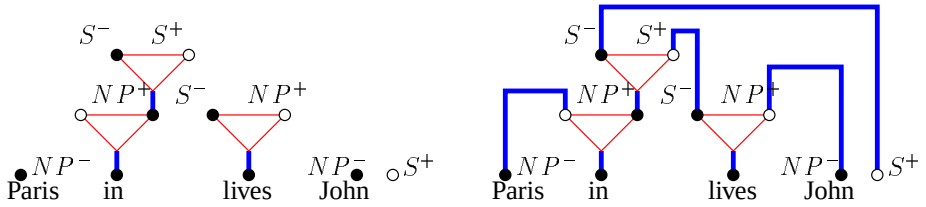
Table 3: Lexicon

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

2.3 Semantic Proof Nets

In this section, we present how (de Groote and Retoré, 1996) propose to use proof nets as semantic recipes. As a slight difference with this work, we only deal in this paper with semantic recipes that correspond to *linear* λ-terms in the Montague's semantics framework.

The idea of expressing the semantics with proof nets refers to the fact that both the λ-terms (with the Curry-Howard isomorphism) and the 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 (de Groote and



(a) Unfolding of the syntactic types

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

Figure 1: Parsing of *John lives in Paris*

Retoré, 1996), based on (Lamarche, 1995)'s dependency paths), we can obtain a λ-term.

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. Of course, we want to keep the compositionality principle of Montague's semantics that maps any syntactic association rule with a semantic association rule. We express it in a straightforward way with the following homomorphism (for as many basic categories as required):

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

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.

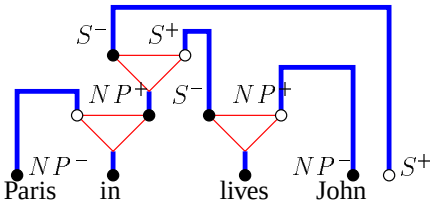


Figure 2: Syntactic proof net for *John lives in Paris*

Let us illustrate the process on a short example. We use the lexicon of table 4 to parse the sentence *John lives in Paris*. The first thing is to define with the syntactic categories of the different lexical items the syntactic proof net of figure 2. It provides the way we should compose the semantic recipes of each lexical item: we take its homomorphic image as in figure 4(a), and we substitute to every input its semantic definition with cut-links.

Then the cut-elimination on the resulting proof net gives a new proof net (on figure 4(b)) we can use as the *semantic analysis* of *John lives in Paris*. If necessary, we can come back to the λ-term expression:$(\textbf{in p})(\textbf{live j})$.

3 Generation: Stating the Problem

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 yields a correct (syntactic) proof net;
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:

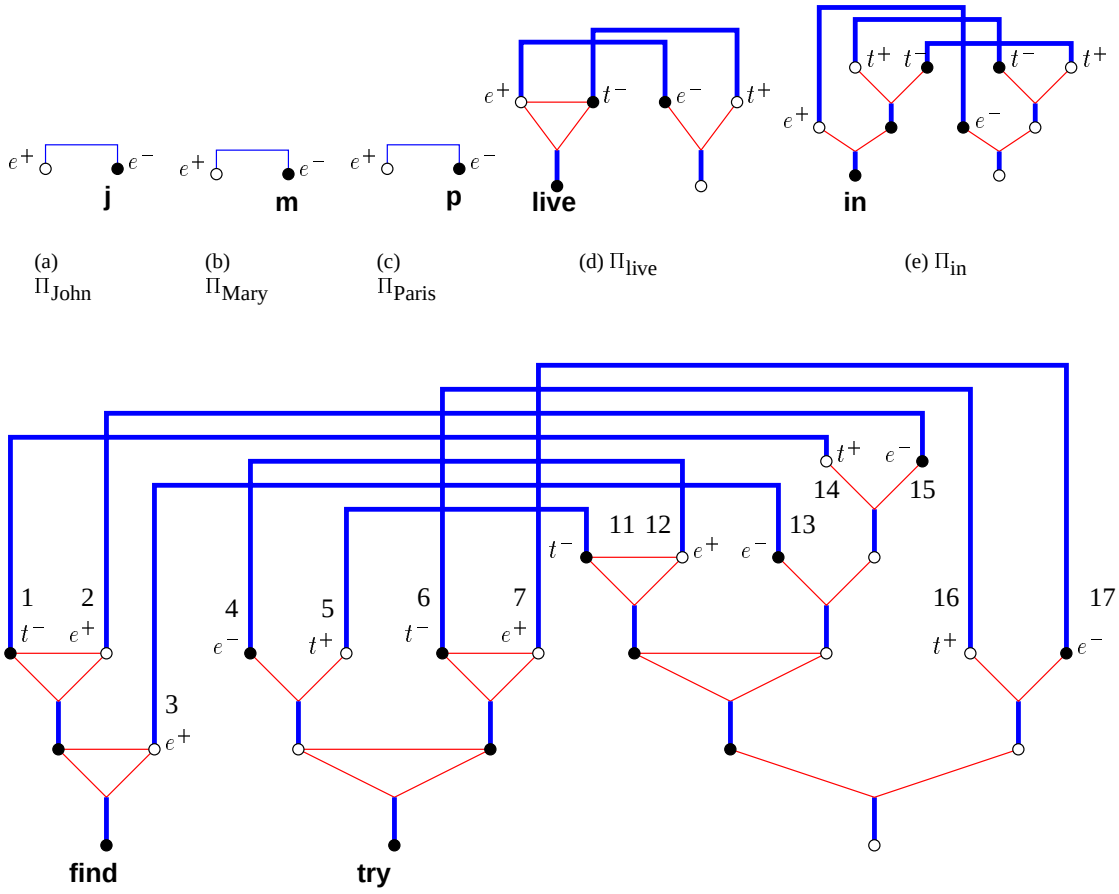
1. F endowed with M (let us call this proof structure F') is a correct proof net;
2. when cut-linking $H(F')$ with the Π_i, and eliminating these cuts, we obtain Π_0.

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 an answer to this problem would consist in taking F and try every possible matching. This brute-force technique 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 an 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 presence of a free variable). And we want to keep these good properties in our algorithm.

Table 4: Lexicon

lexical entry	syntaxic category	associated λ-term	semantic proof net
John	NP	**j**	Π_{John} (cf. figure 3(a))
Mary	NP	**m**	Π_{Mary} (cf. figure 3(b))
Paris	NP	**p**	Π_{Paris} (cf. figure 3(c))
Lives	$NP\backslash S$	$\lambda x.$**live** x	Π_{live} (cf. figure 3(d))
In	$(S\backslash S)/NP$	$\lambda x.\lambda y.$(**in** x)y	Π_{in} (cf. figure 3(e))
Seeks	$(NP\backslash S)/(((NP\backslash S)/NP)\backslash(NP\backslash S))$	$\lambda x.$(**try**(x**find**))	(cf. figure 3(f))



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

(f) Π_{seeks}

Figure 3: Semantic proof nets of the lexicon of table 4

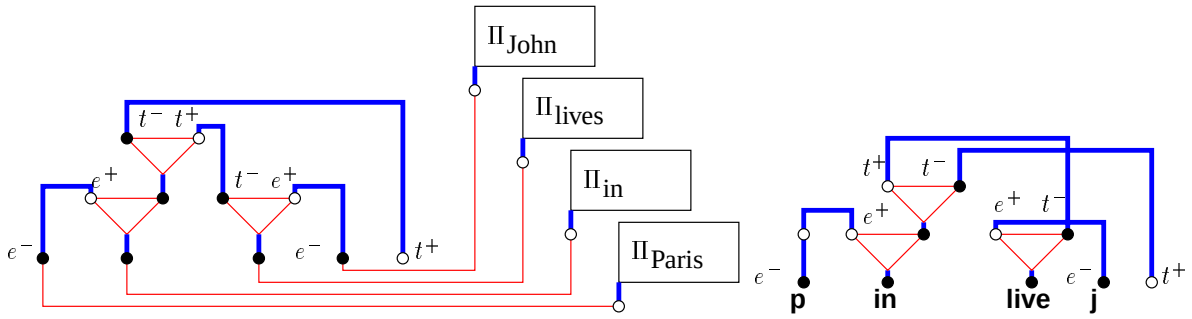
4 Cut-elimination as Matrix Computation

This section first establishes some equivalent relations between cut-elimination on proof nets and matrix equations. We then show how to use these equations in the generation process and how we can solve them. It enables us to characterize the properties required by the semantic proof nets to have a polynomial resolution of the generation process.

4.1 Principles

First, as expressed in (Girard, 1989) and reformulated in (Retoré, 1990; Girard, 1993; Girard, 1995), we state the algebraic representation of cut-elimination on proof nets. Due to lack of space, we can not develop it, but the principle is to express cut-elimination between axioms with incidence matrices and paths in graphs.

Let us consider 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 (we assume without loss of generality that they happen only between axiom links), 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 $\overline{U}$.



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

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

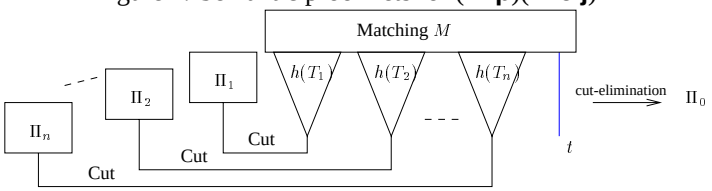


Figure 5: The generation problem

Then we have (Girard, 1989):

$$\Pi = (1-\sigma^2)U(1+\sum_1^\infty(\sigma U)^k)(1-\sigma^2) \qquad (1)$$

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

4.2 Matrix 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 $\overline{U}$ (this assumption entails no loss of generality).

4.3 Expressing the Reduction of U into Π

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

As mentioned in section 4.2, 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 4.2, $B(e_i)$ 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 $p = m$.

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

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):

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

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

If we define $A = ({}^tU_1\Pi_1 - \sigma_1\,{}^tU_1)U_1$ and $X = U_3(1-\sigma_4U_3)^{-1}$, we 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*$(\sigma, U) = (1-\sigma^2)U(1-\sigma U)^{-1}(1-\sigma^2)$
- $({}^tU_1\Pi_1 - \sigma_1\,{}^tU_1)U_1 = \sigma_2U_3(1-\sigma_4U_3)^{-1}\,{}^t\sigma_2$
- $A = {}^t\sigma_2 X \sigma_2$ *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.

4.4 Solving the Equations

In this section (proof search oriented), we consider that the axiom links we are looking for are those whose two conclusions are involved in cut links. That is we want to complete U_3. As in the previous section we proceeded

by equivalence, solving the equation (1) correponds to solving the equation

$$A = \sigma_2 X\, {}^t\sigma_2 \tag{2}$$

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

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

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

Let $\sigma_2 \in \mathcal{M}_{m,n}(\mathbb{R})$, $X = (x_{i,j}) \in \mathcal{M}_n(\mathbb{R})$ and $A \in \mathcal{M}_n(\mathbb{R})$. Let the two sequences $1 \leq i_1 < \ldots < i_l \leq m$ and $1 \leq j_1 < \ldots < j_l \leq m$ be such that with $(a,b) \in [1,m] \times [1,n]$, $E_{ab} = (\delta_{ia}\delta_{jb})_{i,j\in[1,m]\times[1,n]}$[2]

$$\sigma_2 = \sum_l E_{i_l j_l} = \sum_l (\delta_{i i_l}\delta_{j j_l})_{i,j}$$

In other words, $\sigma_{2\,ij} = 1 \Leftrightarrow \exists l_1 \in [1,l] \wedge i = i_{l_1} \wedge j = j_{l_1}$.

Then

$$\begin{aligned}\sigma_2 X\, {}^t\sigma_2 &= (\sum_{l_1=1}^{l} E_{i_{l_1} j_{l_1}}) * X * {}^t(\sum_{l_2=1}^{l} E_{i_{l_2} j_{l_2}}) \\ &= \sum_{l_1=1}^{l}\sum_{l_2=1}^{l} (\delta_{i i_{l_1}} x_{j_{l_1} j_{l_2}} \delta_{j i_{l_2}})_{1\leq i,j\leq m}\end{aligned}$$

It follows that if $A = (a_{ij})_{i,j} = \sigma_2 X\, {}^t\sigma_2$ then

$$\forall (l_1, l_2) \in [1,l]^2, x_{j_{l_1} j_{l_2}} = a_{i_{l_1} i_{l_2}}. \tag{3}$$

A consequence of this result is that if $\sigma_4 = 0$, then $l = n$ and *we determine X completely with relation* (3)*, and then the same for U_3*. This configuration correspond to the fact that in the (given) semantic proof nets, no output contains the two conclusions of a same axiom link. In this latter case, the computation is not so simple and should be mixed with word order constraints.

5 Example

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

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) and in Π (figure 7). This can be done by looking at the typed constants decorating the input conclusions (for the moment, we don't have a general procedure in the complex cases).

We also assume in this numbering that we know which of the atoms in $H(F)$ are linked to t^+ (the unique output). In our case where $\sigma_4 = 0$, it is not a problem to make such a statement. In other cases, the complexity would increase polynomially.

Then, the given matrices are:

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

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

So

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

According to the definition of the (i_l) and the (j_l) families such that $\sigma_2 = \sum_l E_{i_l j_l}$, we have:

i_l	3	4	5	2	1	7	9	10
j_l	1	2	3	4	5	6	7	8

Then

$$\begin{array}{lll} x_{5,2} = 1 = a_{1,4} & x_{2,5} = 1 = a_{4,1} & x_{4,3} = 1 = a_{2,5} \\ x_{3,4} = 1 = a_{5,2} & x_{1,7} = 1 = a_{3,9} & x_{7,1} = 1 = a_{9,3} \\ x_{6,8} = 1 = a_{7,10} & x_{8,6} = 1 = a_{10,7} & \end{array}$$

and in this case $\sigma_4 = 0$, so according to the preceeding notes X is completely determined and

$$X = U_3 = \begin{bmatrix}
0&0&0&0&0&0&1&0\\
0&0&0&0&1&0&0&0\\
0&0&0&1&0&0&0&0\\
0&0&1&0&0&0&0&0\\
0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1\\
1&0&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0
\end{bmatrix}$$

We can add this matching to the syntactic forest of figure 8(a) (do not forget that U_3 represents the edges between e_i with $i \in [17,22]$) and obtain on F the matching of figure 8(b).

[2] ${}^2\delta_{ij} = \begin{cases} 1 \text{ if } i = j \\ 0 \text{ if } i \neq j \end{cases}$

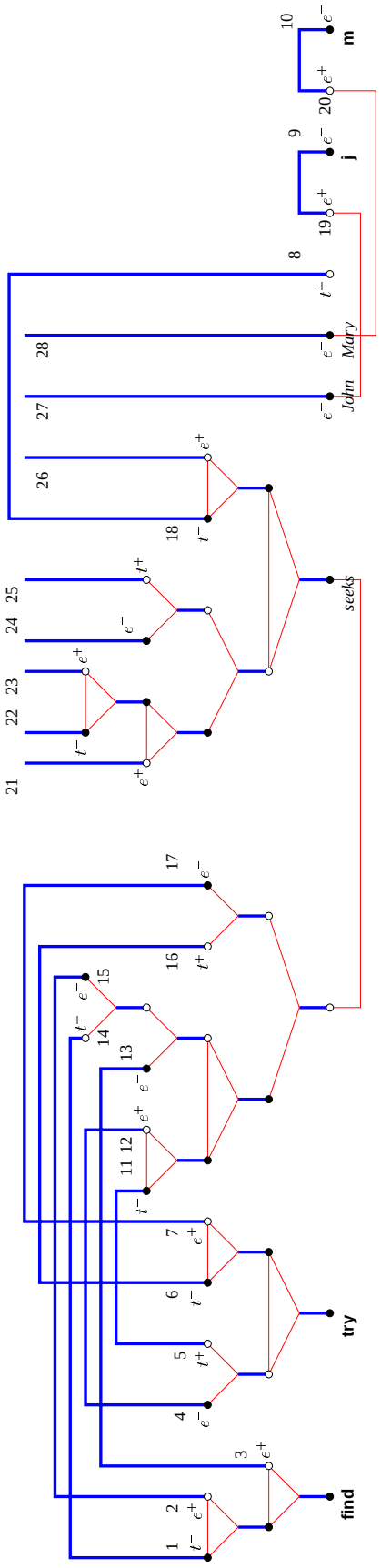


Figure 6: Marking atoms on U

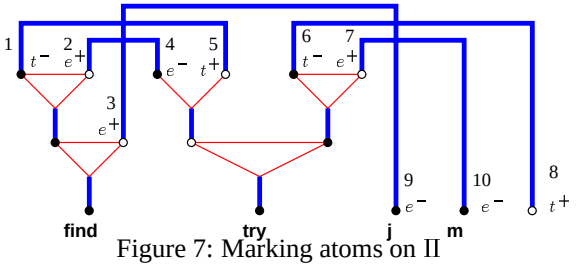
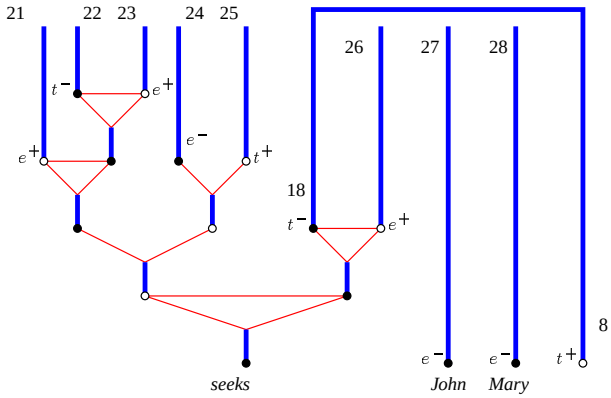
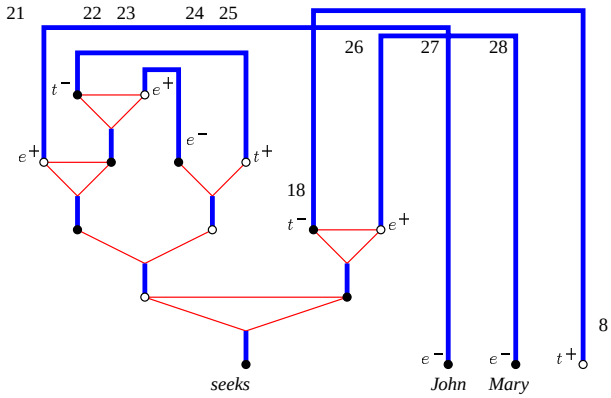


Figure 7: Marking atoms on II



(a) The syntactic forest



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

Figure 8: 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 matrix computation). In this case, it is correct.

Note:

- Actually, this only gives us the axiom links. It still necessitates to compute the word order to have no crossing axiom link. This can be done from the axiom links easier than quadratic time: it is a well-bracketing check. Here, it is easy to see that putting

the *John* item on the left would achieve the result of *Mary seeks John*.

- The choice of *seeks* and its high order type (for intensionnality) shows there is no limitation on the order of the λ-term.

6 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 matrix 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 different extensions of Lambek calculus based on proof nets (Moortgat, 1996; Lecomte and Retoré, 1995) have been considered, 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 Girard's algebraic interpretation of the cut elimination.

References

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

Vincent Danos. 1990. *Une Application de la Logique Linéaire à l'Étude des Processus de Normalisation (principalement du λ-calcul)*. Ph.D. thesis, Université Paris VII, June.

Philippe de Groote and Chritian Retoré. 1996. On the semantic readings of proof-nets. In Glyn Morrill Geert-Jan Kruijff and Dick Oehrle, editors, *Formal Grammar*, pages 57–70, Prague, August. FoLLI.

David R. Dowty, Robert E. Wall, and Stanley Peters. 1981. *Introduction to Montague Semantics*. Kluwer Academic Publishers.

Jean-Yves Girard, Yves Lafont, and P. Taylor. 1988. *Proofs and Types*. Cambridge Tracts in Theoretical Computer Science 7. Cambridge University Press.

Jean-Yves Girard. 1987. Linear logic. *Theoretical Computer Science*, 50:1–102.

Jean-Yves Girard. 1989. Geometry of interaction I: Interpretation of system F. In C. Bonotto, R. Ferro, S. Valentini, and A. Zanardo, editors, *Logic Colloquium '88*, pages 221–260. North-Holland.

Jean-Yves Girard. 1993. Linear logic: Its syntax and semantics. In J.-Y. Girard NS Y. Lafont and L. Regnier, editors, *Advances in Linear Logic*, Ithaca, New York, June.

Jean-Yves Girard. 1995. Geometry of interaction III: The general case. In J.-Y. Girard, Y. Lafont, and L. Regnier, editors, *Advances in Linear Logic*, pages 329–389. Cambridge University Press. Proceedings of the Workshop on Linear Logic, Ithaca, New York, June 1993.

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

François Lamarche and Christian Retoré. 1996. Proof-nets for the lambek calculus – an overview. In V. Michele Abrusci and Claudio Casadio, editors, *Proceedings 1996 Roma Workshop. Proofs and Linguistic Categories*, pages 241–262. Editrice CLUEB, Bologna, April.

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

Joachim Lambek. 1958. The mathematics of sentence structure. *American Mathematical Monthly*, 65(3):154–170.

Alain Lecomte and Christian Retoré. 1995. Pomset logic as an alternative categorial grammar. In *Formal Grammar*, Barcelona.

Josep M. Merenciano and Glyn Morrill. 1997. Generation as deduction on labelled proof nets. In Christian Retoré, editor, *Proceedings of the 1st International Conference on Logical Aspects of Computational Linguistics (LACL-96)*, volume 1328 of *LNAI*, pages 310–328, Berlin, September23–25 . Springer.

Richard Montague. 1974. *Formal Philosophy: Selected Papers of Richard Montague*. Yale University Press, New Haven, CT.

Michael Moortgat. 1996. Categorial type logics. In Johan van Benthem and Alice ter Meulen, editors, *Handbook of Logic and Language*, pages 5–91. Elsevier Science Publishers, Amsterdam.

Christian Retoré. 1990. A note on turbo cut elimination. Manuscript, September.

Christian Retoré. 1993. *Réseaux et séquents ordonnés*. Ph.D. thesis, University of Paris VII.

Dirk Roorda. 1991. *Resource Logics: Proof-theoretical Investigations*. Ph.D. thesis, University of Amsterdam, September.