Generation, Lambek Calculus, Montague's Semantics and Semantic Proof Nets

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

Abstract

Most of the studies in the framework of Lambek calculus have considered the parsing process and ignored the generation process. This paper wants to rely on the close link between Lambek calculus and linear logic to present a method for the generation process with semantic proof nets. We express the process as a proof search procedure based on a graph calculus and the solutions appear as a matrix computation preserving the decidability properties, and we characterize a polynomial time case.

1 Introduction

From the type logical grammars point of view, the parametric part of the language analysis is the lexicon, and the constant one is the logical rules. This should of course hold both for parsing and generation, hence we can consider the *reversibility* properties of such grammars. And a relevant problem is to compare the complexity of the two cases.

For Lambek calculus (Lambek, 1958), the parsing complexity is still an open problem. But the question arises to know *how* to generate in this framework, and *how difficult* (on the computational side) it is. (Merenciano and Morrill, 1997) answered with a labelled deductive system guided with λ-term unification. But a drawback of this latter mechanism consists in its algorithmic undecidability (from second order unification).

Relying on the linear logic (Girard, 1987) (which provides a powerful framework to express Lambek calculus, specially with proof nets for the latter (Roorda, 1991; Lamarche and Retoré, 1996)), this paper wants to adress the problem of finding the way we can associate given lexical entries to fit a given semantic expression and generate a syntactically correct expression (for the moment, we do not care to the choice of the lexical items). 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 syntactic realization process. Then we give an example of this process.

2 Proof Nets for Linear Logic

Linear logic (Girard, 1987) proposes for proofs a more compact and accurate syntax than sequent calculus: proof nets (they group distinct sequential proofs that only have inessential differences). They have both a related to sequential proof definition and a geometrical definition: they can be defined as a class of graphs (proof structures) satisfying a geometrical property so that every proof net corresponds to a sequential proof and every proof structure built from a sequential proof has this property (Retoré, 1998).

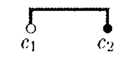
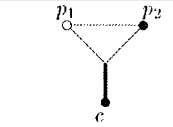
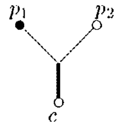
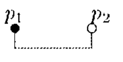
In this paper, we only consider proof nets of the intuitionistic implicative linear logic: sequents are made of several antecedent formulas, but only one succedent formula. To deal with the intuitionistic notion for proof nets (since we consider one-sided sequents), we use the notion of polarities with the *input* (•: *negative)* and the *output* (∘: *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* (we consider implicative fragment) 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.

Proof nets are proof structures that respect the correctness criterion.

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	$c_1 : A^+$ $c_2 : A^-$	$p_1 : A^+$ $p_2 : B^-$ $c : (A \multimap B)^-$	$p_1 : A^-$ $p_2 : B^+$ $c : (A \multimap B)^+$	$p_1 : A^-$ $p_2 : A^+$

The last link of table 1, the Cut link, allows the combination of proofs of $\Gamma \vdash A$ and of $A, \Delta \vdash B$ into a single proof of $\Gamma, \Delta \vdash B$. In sequential calculs, the cut-elimination property states that there exists a normal (not using the Cut rule) proof for the same sequent only from premises of Γ and Δ (and builds it).

Of course, this property holds for proof nets too. And to enforce the intrinsic definition of these latter, a simple rewriting process (described in table 2) actually performs the cut-elimination (in case of complex formulas as in the third rewriting rule, those rules can apply again on the result and propagate until reaching atoms).

2.1 Proof Nets for Lambek Calculus

As Lambek calculus is an intuitionistic fragment of non commutative linar logic (with two linear implications: "\" on the left and "/" on the right), proof nets for it naturally appeared in (Roorda, 1991). They slightly differ from those of table 1:

- we get two tensor links: one for the formula $(B/A)^-$ (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.

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. Proving that *John lives in Paris* is a correct sentence w.r.t. the lexicon of table 3 (the two first columns) is finding axiom links between the atoms in the figure 1(a) so that the proof structure is correct. Figure 1(b) shows it actually happens (for technical reasons, in the proof net, the order of the syntactic categories is the inverse of the order of the words in the sentence to be analysed. Figure 1(c) shows *John lives Paris in* cannot be successfully parsed).

2.2 Proof Nets for Montague's Semantics

Capitalizing on the fact that both λ-terms (with the Curry-Howard isomorphism) and proof nets represent proofs of intuitionistic implicative linear logic, (de Groote and Retoré, 1996) propose to use proof nets as semantic recipes: since proof nets encode linear λ-terms, instead of associating a λ-term in the Montagovian style to a lexical entry, they associate a proof net (decorated with typed constants). An example of such a lexicon is given in table 3[1] (par links encode abstraction, and tensor links encode application).

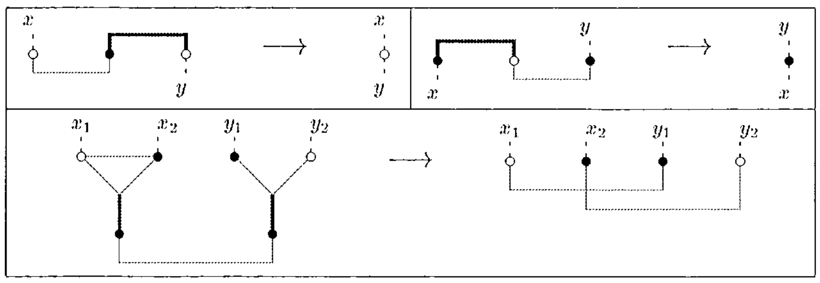
Of course, to respect semantic types based on Montagovian basic types e and t, they use the following homomorphism:

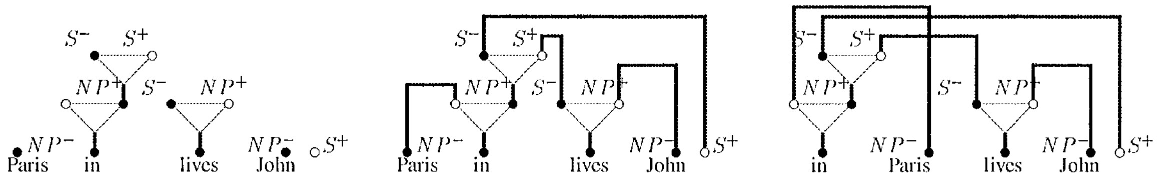
$$\mathcal{H}(NP) = e \quad \mathcal{H}(S) = t \quad \mathcal{H}(A\backslash B) = \mathcal{H}(A) \multimap \mathcal{H}(B)$$
$$\mathcal{H}(N) = e \multimap t \qquad \mathcal{H}(A/B) = \mathcal{H}(B) \multimap \mathcal{H}(A)$$

Let us illustrate the process in parsing the sentence *John lives in Paris*. First we have to find the syntactic proof net of figure 1(b) as explained in 2.1. It provides the way syntactic componants combine, hence how semantic recipes of each lexical item combine: we take its homomorphic image

[1] Unlike in (de Groote and Retoré, 1996), we restrict ourselves for the moment to linear λ-terms.

Table 2: Cut-elimination rewriting rules





S^- S^+ NP^+ S^- NP^+ NP^- Paris in lives NP^- John S^+

(a) Unfolding of the syntactic types

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

(c) Incorrect proof structure for parsing *John lives Paris in*

Figure 1: Parsing of *John lives in Paris*

as in figure 3(a). The substitution of every input with its semantic definition we would like to perform on the λ-calculus side appears on the logical side as plugging semantic proof nets with cut-links.

Then, the β-reduction we would like to perform has its logical counterpart in the cut-elimination on the resulting proof net. It gives a new proof net (on figure 3(b)) we can use as the *semantic analysis* of *John lives in Paris*. If necessary, we can come back to the λ-term expression: $(\textbf{in p})(\textbf{live j})$.

In other words, the syntactic proof net yields a term t expressing how the elementary parts combine (in this case $t = (ab)(cd)$). Then the resulting proof net of figure 3(b) corresponds to the β-normal form of $t[\lambda x.\lambda y.(\textbf{in}\ x)y/a, \textbf{p}/b, \lambda x.\textbf{live}\ x/c, \textbf{j}/d]$.

3 What is Generation?

We can now state the problem we are dealing with: given a semantic proof net (like the one in figure 3(b)), we want to put together 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.

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 (as in figure 1(a));
- F the forest made of the syntactic trees (T_i) of all the considered lexical entries plus the output (the type we want to derive),

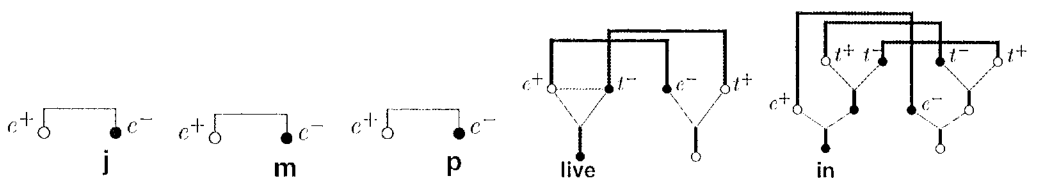
the generation problem (see figure 4) is to find a matching M of atomic formulas of F such that:

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

We note that the problem is intrinsically decidable (because the finitness of the number of the matchings) *without* making any assumption on the form of the semantic entries. Of course, we want to keep these good properties in our algorithm.

Table 3: Lexicon

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



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

Figure 2: Semantic proof nets of the lexicon of table 3

4 Cut-elimination as Matrix Computation

Using proof nets *resulting* from a cut-elimination to guide a proof search on proof nets *before* cut-elimination relies on the algebraic representation of cut-elimination on proof nets expressed in (Girard, 1989) and reformulated in (Retoré, 1990; Girard, 1995). Due to lack of space, we can not developp 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 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 $\overline{U}$. Then we have (Girard, 1989):

$$\Pi = (1-\sigma^2)U(1-\sigma U)^{-1}(1-\sigma^2) \qquad (1)$$

We want to give an equivalent relation to (1) focusing on some axiom links we are interested in. Without loss of generality, we assume the lack of any axiom link in $\overline{U}$ such that none of its conclusions are involved in cut links.

Then we can choose an order for the atoms (from the proof net before the cut-elimination, there is three subsets of atoms: those not involved in a cut link, those involved in a cut link and whose dual is not involved in a cut link, and those involved in a cut link and their dual as well) such that:

$$U = \left[\begin{array}{c:c:c} 0 & U_1 & 0 \\ \hdashline {}^tU_1 & 0 & 0 \\ \hdashline 0 & 0 & U_3 \end{array}\right] \quad \Pi = \left[\begin{array}{c:c:c} \Pi_1 & 0 & 0 \\ \hdashline 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 \end{array}\right] \quad \sigma = \left[\begin{array}{c:c:c} 0 & 0 & 0 \\ \hdashline 0 & \sigma_1 & \sigma_2 \\ \hdashline 0 & \sigma_3 & \sigma_4 \end{array}\right]$$

Note that all the atoms belonging to the matching we are looking for in the generation process (see figure 4) are in U_3.

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:

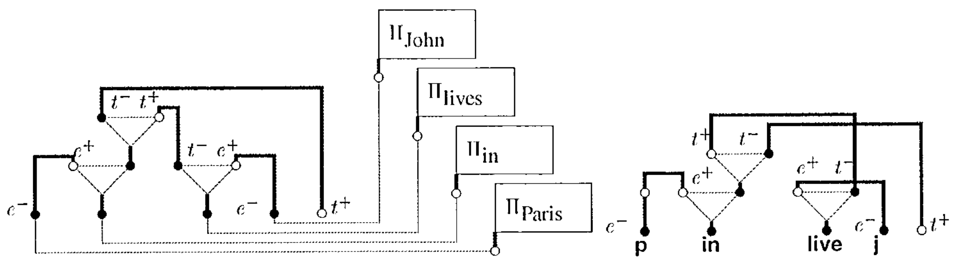
Theorem 1 *Let $\overline{U}$ be a correct proof net reducing in $Res(\sigma, U)$ after cut-elimination. These relations are equivalent:*

- $Res(\sigma, U) = (1-\sigma^2)U(1-\sigma U)^{-1}(1-\sigma^2)$
- $({}^tU_1\Pi_1 - \sigma_1\,{}^tU_1)U_1 = \sigma_2U_3(1-\sigma_4U_3)^{-1}\,{}^t\sigma_2$
- $A = {}^t\sigma_2X\sigma_2$ *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.

Indeed, 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 (knowing all the other matrices). The previous theorem states that solving the equation (1) correponds to solving



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

Figure 3: Semantic proof nets for (**in p**)(**live j**)

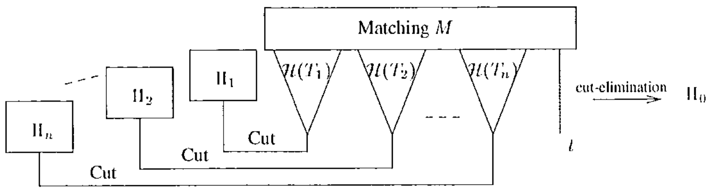


Figure 4: The generation problem

the equation $A = \sigma_2 X\, {}^t\sigma_2$ in X with X inversible. Then, we have to solve $U_3 = X^{-1} + \sigma_4$ such that ${}^tU_3 = U_3$ and $U_3^2 = 1$.

Theorem 2 *If $\sigma_4 = 0$ and there exists a solution, then the latter is unique and completely defined (as matrices product) from A and σ_2.*

If $\sigma_4 \neq 0$ we generally have many solutions, and we have to investigate this case to obtain good computational properties for example in adding word order constraints.

Nevertheless, we can decide the case we are handling *as soon as we are given the lexical entries.*

5 Example

Let us process on an example the previous results. We still use the lexicon of table 3, and we want to generate (if possible) a sentence whose meaning is given by the proof net of figure 3(b).

We first need to associate every atom with an index (in the figures, we indicate a number i beside the atom). Of course, we have to know how to recognize the atoms that are the same in $\overline{U}$ (figure 5(b)) and in $\overline{\Pi}$ (figure 5(a)). This can be done by looking at the typed constants decorating the input conclusions (for the moment, we don't have a general procedure).

We also assume in this numbering that we know which of the atoms in $\mathcal{H}(F)$ is 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 at most polynomially.

Then, the given matrices are:

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

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

And the unique solution is:

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

We can add this matching to the syntactic forest of figure 6(a) (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 atoms with $i \in [17, 22]$) and obtain on F the matching of figure 6(b).

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.

Actually, this only gives us the axiom links. It still requires 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 bracketing problem).

6 Conclusion

We showed that the problem of generation in the Lambek calculus framework is decidable, and we relied on semantic proof nets to express it 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 processing.

Nevertheless, some work remains: we should soon work on the atom numbering and the choice of the lexical items to enable a practical implementation of this work. Moreover, we want to benefit from the power of linear logic (and modalities) to 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, and the anonymous reviewers for their helpful comments.

References

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.

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. 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.

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.

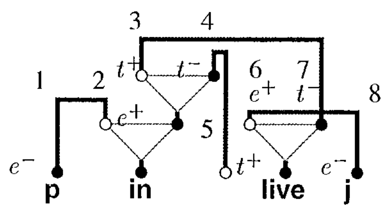
Josep M. Merenciano and Glyn Morrill. 1997. Generation as deduction on labelled proof nets. In Christian Retoré, editor, *Proceedings of LACL-96*, volume 1328 of *LNAI*, pages 310–328, Berlin. Springer.

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.

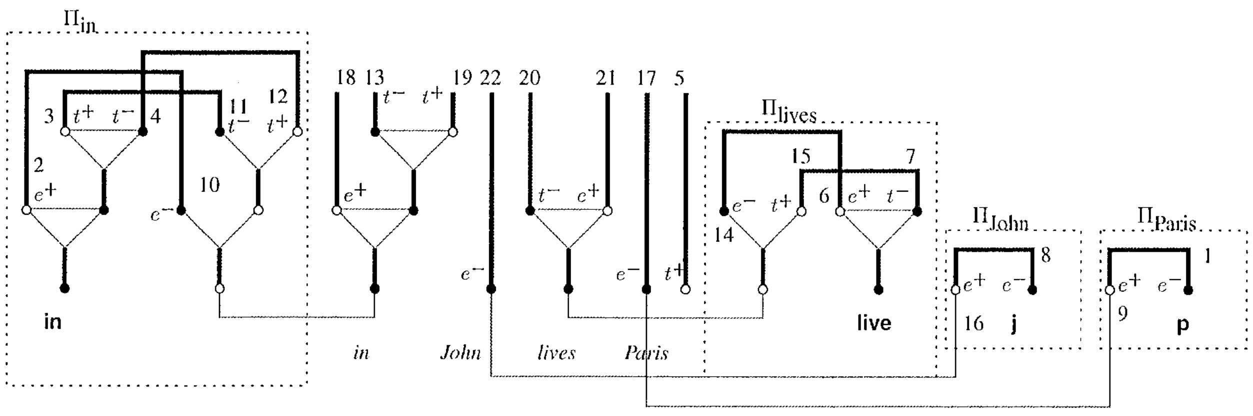
Christian Retoré. 1990. A note on turbo cut elimination. Manuscript, September.

Christian Retoré. 1998. Handsome proofnets: R&b-graphs, series-parallel graphs and perfect matchings. Technical Report 1234, IRISA.

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

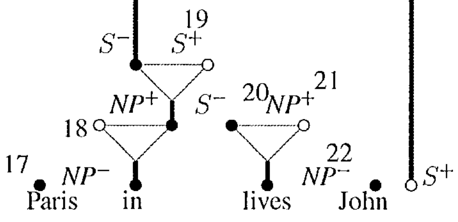


(a) Marking atoms on Π

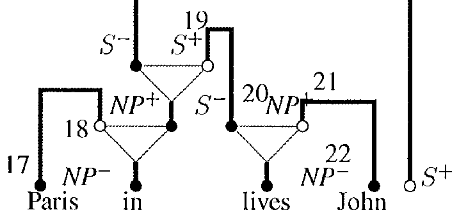


(b) Marking atoms on U

Figure 5: Defining the indices on $\overline{U}$ and $\overline{\Pi}$



(a) The syntactic forest



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

Figure 6: Applying the matching on the syntactic forest