Words as Modules: a Lexicalised Grammar in the framework of Linear Logic Proof Nets

Alain Lecomte[a], Christian Retoré[b]

[a] *Projet Calligramme & Université Pierre Mendès-France, Grenoble*
[b] *Projet Calligramme, INRIA-Lorraine & CRIN-C.N.R.S., Nancy*

Abstract

We present a syntactic calculus relying on the notion of proof net, ie. a geometric representation for proofs in linear logic. The lexicon associates a module or partial proof net with each word, and this information completely encodes the syntactic behavior of the word. Parsing a sentence consists in combining the modules associated with its words into a complete proof net. In order to handle word order, this model takes place in a non-commutative extension of linear logic called Pomset logic.

Introduction

In this paper we describe the principles of a syntactic calculus whose building blocks are partial proof-nets or modules. The main idea is to associate

with each lexical item one or more modules which encode(s) its syntactic behaviour. The simplest of these modules are obtained by unfolding the components of formulae that would be the type(s) of the lexical items in a type-logical grammar à la Morrill (1994), while the more sophisticated ones really go beyond the usual type-logical approach. The syntactic analysis within such a paradigm consists in combining these modules into a complete proof-net by a uniform set of plugging rules.

This approach is related to the Partial Proof-Trees as building blocks of a categorial grammar of Joshi and Kulick (1995,1997), the main difference being the emphasis put on the geometric notion of Proof-Net as in our first attempt (Lecomte and Retoré 1995).

Our main motivation is to obtain a general logical model in which it would be possible to embed other calculi like Lambek grammars on one side and Lexicalised Tree Adjoining Grammars on the other side.

The Lambek calculus is a very elegant syntactic calculus because it is a pure logical calculus enjoying all the properties one can expect: cut-elimination, denotational semantics, truth valued semantics. This is also the reason why it allows a very simple interface with Montagovian semantics. Unfortunately, it suffers from many limitations when applied to linguistic descriptions: for instance it does not handle head-wrapping, cross serial dependencies, right extraction, extraposition from a non-peripheral site etc.

On the other side, the LTAG model provides us with a very efficient model which succeeds in many cases where the Lambek calculus fails. But, because some problems are also unsolvable in L-TAG (like long-distance scrambling in German, Romance Clitics or Kashmiri wh-extraction), variants of TAG have been developed, like Multi-Component TAG of Joshi (1987), Multi-Component TAG with Domination Links of Becker, Joshi and Ranbow (1991) and D-Tree Grammar of Rambow, Vijay-Shanker and Weir (1995). These models have much expressive power, even if they stay in a reasonable range of complexity. But they use an algebraically complex formulation in terms of trees, and as usual with non-logical formalism the relation to semantics is not simple.

In the present paper, we propose a logical system, based on the proof net syntax of Linear Logic which incorporates operations similar to the ones of TAGs as the plugging rules. The key point in these plugging rules is that they preserve a very simple correctness criterion — which states that objects we construct correspond to *proofs* in the underlying logical system.

We first give an overview of the whole logical system, called POMSET-logic, introduced in Retoré (1993) — see Retoré (1997) for an updated presentation in English. Roughly speaking, this logical calculus is based on Multiplicative Linear Logic enriched with the non-commutative connective *before*, and deals with Partially Ordered Multi-sets instead of ordinary multi-sets of formulae. Then, we present a restricted version of the proof nets of Pomset logic. Indeed this restriction is enough for linguistic descriptions, and its linguistic meaning is easier to understand than the very general first attempt of Lecomte and Retoré (1995).

Linear Logic and Word Order

Linear Logic (LL) introduced in Girard (1987) (see Girard 1995 for an excellent survey) is obtained from classical logic by introducing two modalities ? and ! which control the structural rules of contraction and weakening. Thus the ordinary connectives *or* and *and* split into a *multiplicative* — respectively denoted by $\wp$ and $\otimes$ — version and an *additive* one — respectively denoted by $\oplus$ and $\&$. As this system is classical it involves an involutive negation, denoted by $(...)^{\perp}$ which is both an additive and a multiplicative connective. The multiplicative connectives handle the notion of resource management, while the additive ones describe choices.

Our work takes place into the multiplicative fragment, which is simpler and convenient for our purpose — in particular it does not allow any form of contraction or weakening. This logic is classical, in the sense that negation is involutive $(F^{\perp})^{\perp} = F$, and $\wp$ and $\otimes$ are the dual one of the other via the following de Morgan rules: $(F \wp F')^{\perp} = F^{\perp} \otimes F'^{\perp}$ and $(F \otimes F')^{\perp} = F^{\perp} \wp F'^{\perp}$. As usual in the classical case, there is an implication defined by means of negation and disjunction: $A \multimap B = A^{\perp} \wp B$.

In order to handle word order, one needs some non-commutativity. If the two connectives are non-commutative, i.e. if the structural rule of exchange is left out like in Abrusci (1991) one gets the classical extension of the Lambek calculus.

Our approach within the framework of Pomset Logic is completely different: we enrich (commutative) MLL with a non-commutative multiplicative (but still associative) connective called *before*. The de Morgan rules extend to this connective, which is self-dual — $(A < B)^{\perp} = A^{\perp} < B^{\perp}$ — and which

lies in between $\otimes$ and $\wp$ wrt. linear implication: $A \otimes B \multimap A < B$ and $A < B \multimap A \wp B$. In order to handle this connective, we need to consider that the multiset of conclusions of a proof is partially ordered — in general, a proof has several conclusions in a classical system. This is particularly suitable in natural language processing to handle relatively free word order.

For people familiar with the Lambek calculus we insist that $A^\perp < B$ is very different from $A \multimap B = A^\perp \wp B$. Indeed, the order it introduces as little to do with the order in the Lambek calculus: for instance A and $A^\perp < B$ leads to B only if there is no order between A and $A^\perp < B$.

Proof nets for Pomset Logic

As a consequence of the afore mentioned de Morgan laws we can restrict ourselves to formulae in which negation is only applied to propositional variables. Thus the language that we consider is defined by:

$\mathcal{P}$ a set of propositional variables — positive atoms

$\mathcal{P}^\perp$ the set of negated propositional variables — negative atoms

$\mathcal{F} ::= \mathcal{P} \mid \mathcal{P}^\perp \mid \mathcal{F} \wp \mathcal{F} \mid \mathcal{F} < \mathcal{F} \mid \mathcal{F} \otimes \mathcal{F}$

Beware that when we write $G^\perp$, this is just a short hand for the unique formula of this language which is obtained from $G^\perp$ by internalising the negation until it reaches the propositional variables, by means of the de Morgan laws.

Exactly like the best way to represent proofs in Intuitionistic Logic consists in using Natural Deduction (where proofs look like (pseudo-)trees), the natural syntax for Linear Logic (LL) is proof net syntax that was introduced in the original paper (Girard, 1987).

As shown in Retoré (1993) Pomset logic very simply extends the usual proof net syntax and our presentation will follow Retoré (1997). It is based on the notion of R&B-graph.

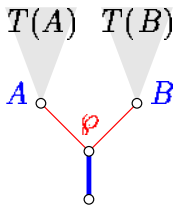
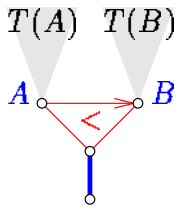
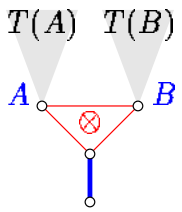
Let us call a **R&B-graph** an edge bicoloured graph, the two colours of the edges being B(bold, blue) and R(regular, red). The B-edges are undirected and never adjacent. The R-edges may be undirected or directed, in which case we call them R-arcs. They correspond to connections between formulae (connective-links), or to the partial order between conclusions.

Given a formula F, we define here its **R&B-tree** $T(F)$, as an inductively defined R&B-graph. Actually it is not a tree, but looks like the sub-formula tree, hence the name *tree*.

Given an atomic formula c, the discrete graph is the R&B-tree of C, with root c, and leaf c.

If C is a compound formula $A * B$ with $* \in \{\wp, \otimes, <\}$ and if $T(A)$ and $T(B)$ are respectively the R&B-trees of the formulae A and B, Table 1. describes, according to what $*$ stands for, a R&B-tree of C, with root C, and both the leaves of $T(A)$ and $T(B)$ as leaves.

Table 1.

Formula C	$A \wp B$	$A < B$	$A \otimes B$
R&B-tree $T(C)$			

The added edges, arcs and vertices are called a $*$-link: A and B are said to be its premises and $A * B$ its conclusion. Notice that each bottom vertex of a B-edge of the R&B-tree $T(C)$ of a formula C, is labelled with a sub-formula of C. Out of these R&B-trees we define the R&B-proof-net, whose building blocks are links.

A **partial proof structure** with conclusions and cuts $C_1, \ldots, C_n$ and cuts $G_1^\bullet, \ldots, G_p^\bullet$ — ie. formulae $G_i = X_i \otimes X_i^\perp$ marked as cuts — with order $\Re$ on its conclusions and cuts is a R&B-graph which consists in:

- the family of R&B-trees $T(C_1), \ldots, T(C_n), T(G_1), \ldots, T(G_p)$. This part represents the *syntactic forest*, of the sequent $C_1, \ldots, C_n, G_1, \ldots, G_p$
- a family of B-edges, called *axioms*, each of them linking two leaves being the negation one of the other, in such a way that
  (*) for each leaf there exists at most one axiom incident to it.
- a family of R-*arcs*, representing the *order*, i.e. there is one such R-arc from a conclusion or cut X to another Y whenever $X < Y[\Re]$

– a special mark, $\bullet$, on the roots of the $T(G_i)$ to make a distinction between a cut and a conclusion $X \otimes X^\perp$ which is not considered as a cut.

The leaves which are not incident to any B-edges are called the **hypotheses** of the (partial) proof structure.

A **proof structure** is a partial proof structure with no hypothesis — in this case, there is a B-edge incident to each vertex. Given a proof structure a conclusion is called an **output** whenever it is the positive conclusion a of an axiom $a, a^\perp$ whose negative conclusion $a^\perp$ is the premise of a *times* link. A proof structure is said to be **intuitionistic** whenever it has a single output.

A (partial) proof structure is said to be a **(partial) proof net** whenever is does not contain any alternate elementary (æ) cycle, that is to say a cycle consisting in a sequence of consecutive edges which are alternatively B and R edges in which each vertex appears exactly once.

We will mainly need the following result, which applies to partial proof nets provided a cut does not reach an hypothesis — clearly, in this latter case, the cut can not be eliminated. The calculus of ordered proof nets enjoys cut-elimination, which is strongly normalising and confluent: a proof net with conclusions and cuts $F_1, ..., F_n, G_1^\bullet, ..., G_p^\bullet$ ordered by $\Re$ reduces to a cut free proof net with conclusions $F_1, ..., F_n$ ordered by $\Re|_{F_1,...,F_n}$.

As we do not use proof nets with an order on the conclusions and cuts, the algorithm performing cut-elimination may be described as follows:

– a cut link between $A * B$ and $A^\perp \star B^\perp$ where $(*, \star) \in \{(<, <), (\wp, \otimes), (\otimes, \wp)\}$, is erased and replaced with two cut links, one between A and $A^\perp$ and one between B and $B^\perp$

– a cut link between a and $a^\perp$ with $a \in \mathcal{P}$ is suppressed, as well as the two vertices a and $a^\perp$ and their incident B-edges — the two axioms which they are respectively the conclusions of — and a new axiom is added between the $a^\perp$ and a which were previously respectively linked to a and $a^\perp$.

As we already said, the partial proof nets correspond to words and parsing a sentence or a phrase consists in plugging them in order to obtain a proof net. Here are the two ways of plugging two PPN M_1 and M_2 together that we consider:

axiom plugging consists in identifying an hypothesis a of M_1 and a conclusion of a of M_2. This corresponds to complementation according to the input or hypotheses sub-categories.

cut plugging consists in linking by a cut-link two dual conclusions and then performing cut-elimination. This corresponds to modification or adjunction.

Actually we shall only make use of the second construction when one of the dual formulae is of the shape $x \otimes x^{\perp}$ with x atomic, ie. when one of the two may be read as a cut on an atomic formula. Such a conclusion/cut will be call a gate.

Labels for Expressing and Computing Word Order

Now that we have set the general formalism, we will distinguish, among all possible partial proof nets, the ones which deserve a specific linguistic interest. In our system, each word is associated with a partial proof-net. This partial proof-net displays the parts of a formula which expresses the type of the word, that means : *what is its category and how it must be processed in any context where it occurs*. Vertices at the bottom of a B-edge of a partial proof-net are themselves typed with a sub-formula of the conclusion and will be labelled with a partial order on strings. These labels will be given for partial proof nets in the lexicon and computed for compound partial proof nets, by rules of unification in the axiom links, and rules of propagation into the links of partial proof nets.

A partial proof net is correctly labelled whenever each (sub)formula is labelled with a pomset of strings which fulfils the following criteria:

- **Labelled axiom-link** : it has two conclusions, $(A, \mathfrak{R})$ and $(A^{\perp}, \mathfrak{R})$ where $\mathfrak{R}$ is a partially ordered multi-sets of strings — the two labels of the conclusions of an axiom link are equal.

- **Labelled tensor-link** : if the conclusion is labelled by the pomset $\mathfrak{R}$, one of the two premises (we do not know which one) of the tensor-link is labelled with the union of $\mathfrak{R}$ with the pomset $\mathfrak{R}'$ that labels the other premise (thus, if $\mathfrak{R} = \emptyset$, then the two premises are labelled by the same pomset)

- **Labelled before-link** : the conclusion $A<B$ is labelled by the pomset $\mathfrak{R} < \mathfrak{R}'$ (where $<$ means that all the elements of $\mathfrak{R}$ are before all the elements of $\mathfrak{R}'$).
- **Labelled par-link** : the conclusion $A \wp B$ is labelled by the pomset $\mathfrak{R} \cup \mathfrak{R}'$ (no order between elements of $\mathfrak{R}$ and elements of $\mathfrak{R}'$).
- **Labelled cut-link** : the cut is labelled by $\emptyset$ and the pomsets of the two premises are equated (particular case of the tensor-link).

This definition, is a particular case of the order on axioms we used in Lecomte and Retoré (1995) and in the restricted case that we present here the way the order is computed is clearer. Let us justify this labelling:

- Only the *before* connective creates an order on strings.
- The *times* connective preserves the order. Indeed, in a calculus of right handed sequents, a $\otimes$-formula is the dual of an implicative formula, and thus no order is added, since this connective describes a transformation : a sequence of types gives a new type, but the underlying string remains unchanged. Notice that in such a classical system all cases of *par* formulae can be viewed as implicative ones, because of the duality and the De Morgan laws. For instance : $A \wp B = A \multimap B^{\perp}$.
- A $\wp$-formula corresponds to the dual of a $\otimes$-formula. It is therefore natural to retrieve the orderings which occur on the two components, but without creating any new ordering.

From now on we shall call Labelled partial proof nets the ones which are labelled according to the principles we gave above.

It should be noticed, in the previous definition, that the labels may be computed from the (Partial) Proof Nets — but this redundant information is useful to underline word order and to explain how orders compose when plugging Partial Proof Nets.

Lexicalised Intuitionistic Labelled Partial Proof-Nets

Some labelled partial proof nets deserve a special interest because they roughly correspond to the elementary trees of TAGs.

An Intuitionistic Labelled PPN is said to be **lexicalised** if and only if :

1. it has two conclusions: its output b connected by an axiom link to a $b^\perp$ in some Y_i, and its **main conclusion** which is of the following form: $a^\perp \wp (X_1 \otimes Y_1) \wp \ldots \wp (X_m \otimes Y_m)$,
2. $a^\perp$ is labelled by a pomset consisting in one word/string w (which belongs to the lexicon), and the formulae $X_i \otimes Y_i$ by $\emptyset$,
3. $a^\perp$ is linked by an axiom-link to an atom a which occurs in one of the X_i,
4. every X_i is a *par/before* formula, ie. does not contains any *times* , connective.

Let us comment this definition:

- If we remember that connectives on the right correspond to their duals on the left, we see that such a conclusion, when moved to the left, leads to a formula : $a \otimes (X_1 \multimap Y_1^\perp) \otimes \cdots \otimes X_m \multimap Y_m^\perp)$.
- Thus the second condition in the definition makes an occurrence of $a^\perp$ necessary in one of the X_i. This can be interpreted as : the word w is an a-phrase *and* — to be understood as *times*, ie. multiplicative conjunction — when supplied with an X_i-phrase, it yields an $Y_i^\perp$-phrase.
- notice this definition is coherent with our definition of labelled links: only the *declarative* part (w *is* an a-phrase) provides an initial pomset. The *procedural* part contains initially no pomset, but in order to fulfill the principles of the labelling in links, at least one of the X_i will receive a pomset (which contains at least w, because of the axiom link between $a^\perp$ and a, (and because X is a *par/before* formula)), and this pomset will be transfered to $Y_i^\perp$, and possibly to the output (if there are axiom links between the Y_j and the X_k).

Following these intuitions the lexical item *loves* has the following type:

$$(\{loves\} : v) \otimes ((v < np) \multimap vp) \otimes ((np < vp) \multimap s)$$

which is the type of a transitive verb. It gives a string labelling the verb, and two other informations: if followed by an np, it will give a vp and if this

vp is preceded by an np it will give an s. Of course, we need to express the identity of the two occurrences of v and the identity of the two vp. There, the proof representation is needed, since the only possibility of modelling this identity between two nodes by purely logical means is to link them via an axiom link — (cf. Figure 1.).

Figure 1.

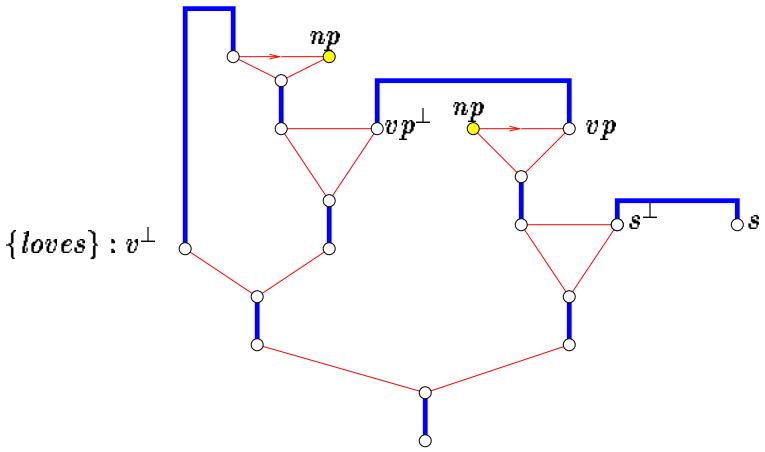
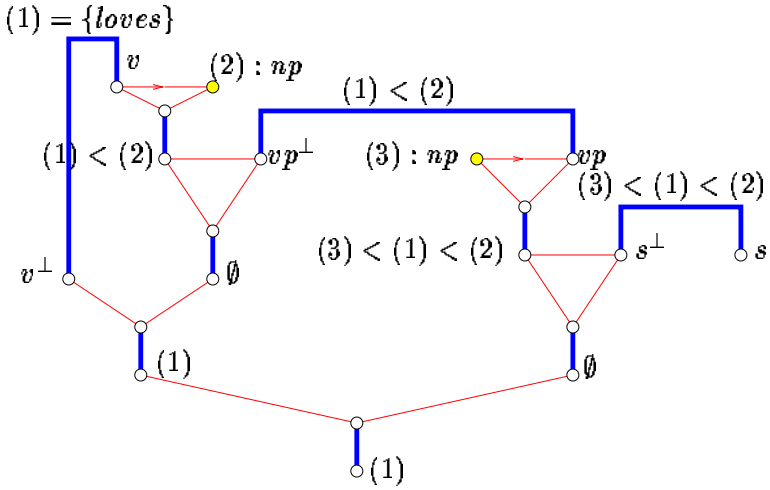


Figure 2.



In Figure 2, we compute, following the rules for labelling, the label of each (sub)formula — labels are denoted by integers. Let us explain how

the labelling rules work on this particular example. By the conditions on labelled links, (1) is identified with {loves}. When the np (2) is identified with a real np, thus giving (2) = $\{a < woman\}$ for instance, the conclusion of the before-link will be labelled by : $\{loves < a < woman\}$. This pomset is transmitted to the output $vp^{\perp}$ (because of the tensor-link). When the second np (3) is instantiated (for instance by $\{a < man\}$), we shall get as a new pomset labelling the conclusion of the second before-link : $\{a < man < loves < a < woman\}$, and this pomset is transmitted to the output $s^{\perp}$. Notice that the conclusions in the $\otimes$-link remain empty:thus the conclusion of the lexicalised PPN associated with the verbal form *loves* is also labelled by $\{loves\}$.

This makes sure that plugging by axiom every hypothesis a with the output a of an a phrase yields the correct word order.

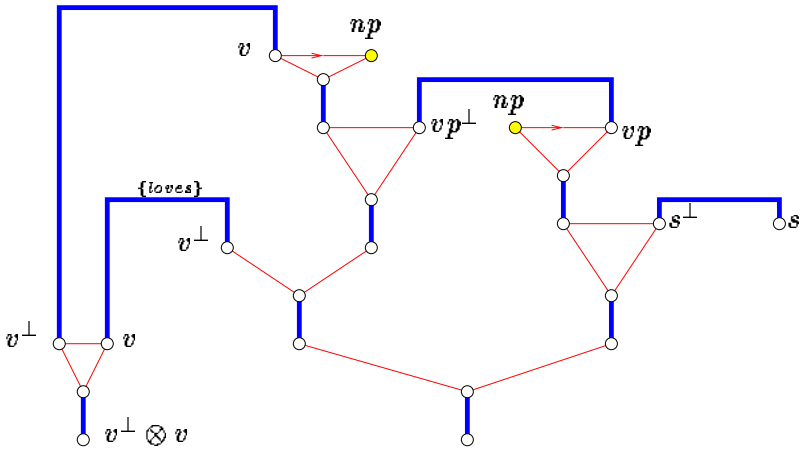
Gates and Modifiers

Up to this stage, the operation of plugging by cut-links can hardly be used: for the time being, there is no obvious possibility of finding interesting cut formulae. But we may consider gates in order to use them and to model modifiers. A gate is a fake conclusion $a \otimes a^{\perp}$ with a being an atomic formula, such that a (resp. $a^{\perp}$) is linked with an axiom link to an $a^{\perp}$ (resp. a) — notice that the absence of æ-cycle makes sure that the two axioms a and $a^{\perp}$ are the conclusions of are distinct. If we were to reduce this cut, these two axioms and the cut would be replaced with a single axiom linking their dual counterpart. The label of a gate will always be empty, as opposed to the main conclusion which is always labelled, like we previously saw, by the same pomset as the *declarative* B-edge. We shall add such gates along the declarative B-edge of a lexicalised labelled partial proof net.

The only modification consists in replacing an axiom link with a sequence axiom link, cut-link, axiom link, and it is easily observed that such a transformation preserves the property of being an ILPPN.

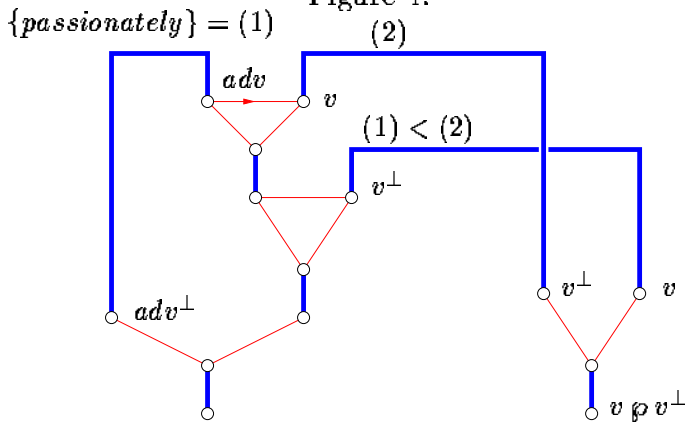
For instance, if we want to introduce adverbials, like *passionately*, we must open such a *gate* in every LPPN associated with a verb. This will give raise to new LPPNs like shown in Figure 3.

Figure 3.



Such an ILPPN can be then used together with a module which introduces a verb modifier. This modifier LPPN contains a dual conclusion $v^{\perp} \wp v$ (cf. Figure 4). It is then possible to link the two modules by means of a cut. Then, by cut-elimination, two axiom-links will be inserted between dual v-nodes one belonging to the adverbial module and the other to the verbal one. The label of the v node, and then the label of the $vp^{\perp}$ node will be transformed and give a correct word order.

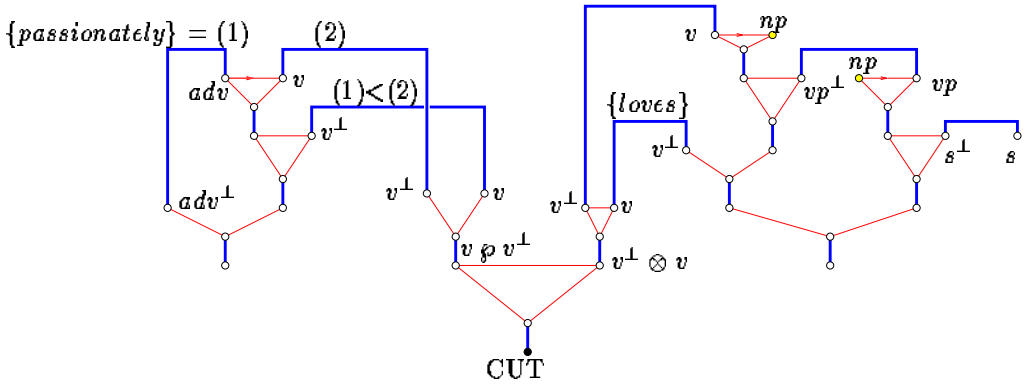
Figure 4.



This leads us to the following definition. A **Modifier-LPPN** is a Labelled Partial Proof-Net with two conclusions, where one of the two conclusions is the *par* of two dual atoms x and $x^{\perp}$, where x is the type of object that the module modifies, and the other one the main conclusion.

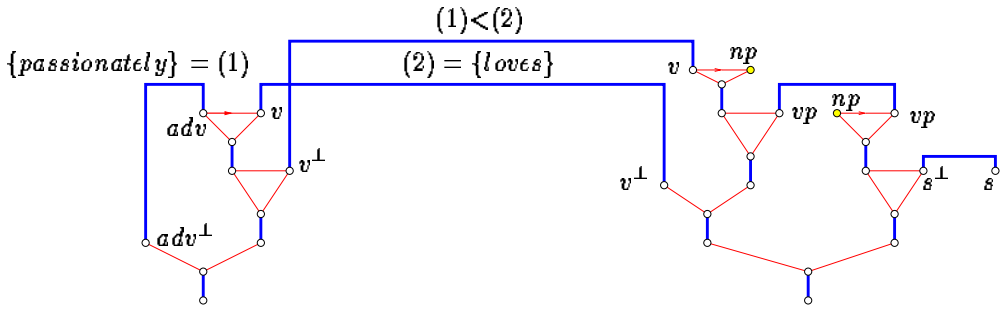
Let us explain the behavior of the MLPPN of Figure 4. When linked by a cut link to an LPPN associated with a verbal form (cf. Figure 5), the corresponding *times* formula is labelled by the same pomset (from the definition of labelled cut-links) and so, the highest v-entry in the verbal PPN will be labelled by $\{passionately, loves\}$, but without any ordering on it. It is the operation of cut-elimination which will provide the correct word order.

Figure 5.



This operation (which is an intrinsical part of the process of plugging by cut) gives us finally the PPN of Figure 6.

Figure 6.



If there is no modifier, the PPN of Figure 3 is still valid. In this case, cut-elimination amounts to suppressing the *times*-link $v \otimes v^{\perp}$, which is, as already said, a mere instance of a cut. Let us notice that without cut-elimination, the word order would remain free. It is a question whether we

could represent in this way dislocated sentences, sentences that result from different orderings of words with particular intonations.

Dicontinuous Constituents

In order to give an example of the expressive power of our system let us consider an example of a construction which is not handled by the Lambek calculus. Let us consider French negation, which is a discontinuous constituent, namely French negation which we will assume to be *ne ... pas* which wraps the verbal head — the variants *ne ... plus*, *ne ... jamais* etc. being completely similar.

The idea is to associate with this negation a MLPPN whose main conclusion is $\{ne\} : neg^{\perp} \wp (\{pas\} : neg \wp ((neg < (v < neg^{\perp})) \otimes v^{\perp})$ and whose secondary conclusion is $v^{\perp} \wp v$.

Let us represent the PPN by a linear expression whose atoms linked by an axiom link are marked by a same label $[i]$, where i is an integer. Then *ne ... pas* is associated with : $\left| \begin{array}{l} [1] : (\{ne\} : neg^{\perp}) \wp (([2] : \{pas\} : neg) \wp \\ (([1] : neg < ([3] : v < [2] : neg^{\perp})) \otimes [4] : v^{\perp})), \\ ([3] : v^{\perp}) \wp ([4] : v) \end{array} \right.$

The dual formula of the main conclusion (or type of the expression) is :

$([1] : \{ne\} : neg)$
$\otimes(([2] : \{pas\} : neg^{\perp}) \otimes (([1] : neg < ([3] : v < [2] : neg^{\perp})) \multimap [4] : v))$

The underlying formula expresses that this module provides the context with *ne* and *pas* — whose types are the dual of the other. The axiom links of the module express that a verb v ($[3]$) wrapped by *ne* and *pas* gives a new verb ($[4]$).

By plugging by cut-link this module with a verbal ILPPN, we obtain the right word order for the negative sentence, for instance *Pierre* **ne** *regarde* **pas** *Marie* (Peter does not look at Mary) thus obtaining a simple logical formulation of head-wrapping, as opposed to hitherto considered extensions of the Lambek calculus.

Conclusion

The system we have presented here is specifically a *resource-conscious sys-*

tem. Duality must be interpreted in terms of exchange and communication. Firstly, the plugging of two atomic formulae via an axiom link expresses that some demand of a phrase is satisfied. Secondly, the plugging via a cut-link makes a communication between two processes which can be synchronised by removing the cut — observe that in this case as well, one cut-formula provides what the other formula (the dual one) needs.

This characteristic allows to express many phenomena. For instance, in wh-extraction as well as in cliticization, a word (the wh-word or the clitic word) is supposed to suppress a demand for an *np* of a given form (for instance an accusative *np* for *what*, or an *np* of the same case of the clitic in the clitic-case). But to suppress a demand is... to satisfy it, just because of the involutive negation — $(x^{\perp})^{\perp} = x$, i.e. a demand for a demand for a x is equivalent to an x.

This kind of mechanism is typical of a system based on resource consumption as may be observed in categorial grammar, for instance in the type-raising operation. Here, we go beyond this because as opposed to the Lambek calculus which only has an implicit non-involutive negation, we take advantage of the explicit and involutive negation of linear logic to extend this phenomenon to other situations, and this results in an important expressive power.

Finally, let us notice the important advantage of this system, being based on logic. We can avoid *ad hoc* and very particular algebraic operations, for instance on trees or strings. Here all the machinery may be summarised as follows: *given the initial LPPNs as they are, build a correct proof-net*, the only requirement being that the correctness criterion, namely *no elementary alternate circuit* is preserved during the construction of the proof net. A future work will study the embedding of TAGs and Lambek grammars in this system.

Acknowledgements: Many ideas in this paper come from discussions in the Calligramme group, especially with Philippe de Groote.

References

V. M.Abrusci. 1991. "Phase semantics and sequent calculus for pure non-commutative classical linear logic". *Journal of Symbolic Logic* 56(4), 1403–1451.

T. Becker, A. Joshi and O. Rambow. 1991. "Long distance scrambling and tree adjoining grammars". In 5^{th} *EACL*, 21 – 26.

J.-Y. Girard. 1987. "Linear logic". *Theoretical Computer Science* 50(1):1–102.

J.-Y. Girard. 1995. "Linear logic: its syntax and semantics". In J.-Y. Girard, Y. Lafont and L. Regnier (eds), *Advances in Linear Logic*. (London Mathematical Society Lecture Notes, vol. 222), Cambridge: Cambridge University Press, 1–42.

A. Joshi. 1987. "An introduction to tree adjoining grammars". In A. Manaster-Ramer (ed.), *Mathematics of Language* Amsterdam and Philadelphia: Benjamins, 87 – 114.

A. Joshi and S. Kulick. 1995. "Partial proof trees as building blocks for a categorial grammar". In G. V. Morrill and R. Oehrle (eds), *Formal Grammar*, FoLLI, 138–149.

A. Joshi and S. Kulick. 1997. "Partial proof trees, resource sensitive logics and syntactic constraints". In C. Retoré (ed.) , *Logical Aspects of Computational Linguistics, LACL'96*, (LNCS/LNAI series), Heidelberg and New-York: Springer-Verlag. To appear.

A. Lecomte and C. Retoré. 1995. "Pomset logic as an alternative categorial grammar". In G. V. Morrill and R. Oehrle (eds), *Formal Grammar*, FoLLI, 181–196.

G. V. Morrill. 1994 , *Type Logical Grammar*. Dordrecht and Hingham: Kluwer.

O. Rambow, K. Vijay-Shanker, and D. Weir. 1995. "D-tree grammars". 33^{rd} *meeting of the Association for Computational Linguistics.*

C. Retoré. 1993. "Réseaux et Séquents Ordonnés". Thèse de Doctorat, spécialité Mathématiques, Université Paris 7.

C. Retoré. 1995. "Pomset logic: a non-commutative extension of classical linear logic". In J. R. Hindley and Ph. de Groote (eds). *Typed Lambda-Calculus and Applications, TLCA'97*. (LNCS, vol. 1210) Heidelberg and New-York: Springer-Verlag, 300–318.