On the semantic readings of proof-nets

Philippe de Groote, Christian Retoré

On the Semantic Readings of Proof-Nets

Philippe de Groote, Christian Retoré

INRIA-Lorraine & CRIN–CNRS
615, rue du Jardin Botanique - B.P. 101
54602 Villers lès Nancy Cedex – FRANCE

e-mail: Philippe.de.Groote@loria.fr, Christian.Retore@loria.fr

A la mémoire de
Xavier de Groote

Abstract. The goal of this paper is to demonstrate how the very rich notion of proof-net may be used, in the framework of categorial grammars, as a unique structure that allows the syntactic and semantic aspects of sentence analysis to be unified. We first explain how the intuitionistic multiplicative proof-nets correspond exactly to the so-called linear λ-terms. This allow us to interpret proof-nets not only as syntactic structures but also as semantic readings *à la* van Benthem. Then, we generalize the correspondence between proof-nets and λ-terms to the complete categorial hierarchy, and we show how Montague-like semantics may be handled in this framework.

1 Introduction

If one were to summarize in a few words the logical principles underlying categorial grammars [15, 17, 24], these could well be: *Parsing as Deduction and Grammar Theory as Proof Theory*. Indeed, during the last decade, proof-theoretical investigations of categorial grammars have been extremely fruitful, e.g., [21, 24].

On the syntactic side, Roorda advocates the notion of proof-net as an appropriate parsing structure [21]. Proof-nets are a new proof-theoretic tool introduced by Girard in the framework of linear logic [6]. They allow several proofs of the sequent calculus to be represented by the same structure when they do not differ in an essential way. In this sense, they correspond to unambiguous representations of proofs. Moreover, their nice mathematical theory gives rise to new parsing algorithms [13, 16, 21].

On the semantic side, van Benthem uses the Curry-Howard correspondence as an interface between syntax and semantics [23]. This correspondence, which dates back to the sixties [9] (see also [5, 8]), establishes an isomorphism between natural deduction, on the one hand, and typed λ-calculus, on the other hand:

Natural Deduction	Typed λ-Calculus
Formulas	Types
Proofs	λ-Terms
Proof Normalization (cut elimination)	β-Reduction

In the framework of categorial grammars, a third collumn may be added to this table:

Natural Deduction	Typed λ-Calculus	Categorial Grammars
Formulas	Types	Semantic Categories
Proofs	λ-Terms	Semantic Readings, Semantic Recipes
Proof Normalization (cut elimination)	β-Reduction	Composition of Semantic Recipes

Now, in the framework of linear logic, the essence of the Curry-Howard isomorphism is stated by Girard as follows:

Formulas	Types
Proofs	Proof-Nets
Proof Normalization (cut elimination)	Proof-Net Evaluation

Therefore, by a simple juxtaposition of the two tables above, one sees that proof-nets may play the semantic part that is usually played by λ-terms. In this paper we introduce and illustrate by several examples this new point of view.

The next section gives a brief introduction to the notion of proof-net, and explains how linear λ-terms may be represented as proof-nets by using a notion of polarity. Section 3 shows how to get semantic readings of categorial principles, such Montagovian type raising, directly from a proof-net. In Section 4, we explain how to adapt the notion of proof-net to the different logics of the categorial hierarchy, i.e., the Lambek calculus L, the intuitionistic logic I, and van Benthem's intermediate logics LP, LC, and LPC. Finally, Section 5 provides a short but complete example of a syntactic and semantic analysis based on proof-nets, in the spirit of Montague's PTQ grammar [14].

While this paper tries to be as self contained as possible, we assume that the reader has some familiarity with categorial grammars [1, 17, 18], the Lambek calculus [12, 15, 24], and linear logic [6, 7, 22].

2 Correspondence between Linear λ-Terms and Intuitionistic Multiplicative Proof-Nets

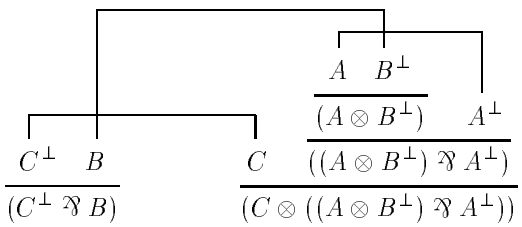
2.1 A Brief Introduction to Multiplicative Proof-Nets

The notion of proof-net has been introduced by Girard [6] as the most suitable way of representing proofs in linear logic. With respect to sequential derivations, proof-nets have at least two advantages: firstly, they are more compact; secondly, they allow sequential proofs that differ in an inessential way to be identified.

Roughly speaking, proof-nets are obtained from sequential derivations by considering only the active formulas and by linking together the formulas that occur in the same axiom. Consider, for instance, the following derivation where the active formulas are framed:

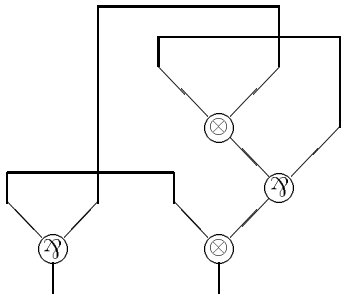
$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash A^{\perp}, A} \quad \dfrac{}{\vdash B^{\perp}, B}}{\vdash B, \boxed{(A \otimes B^{\perp})}, A^{\perp}} \quad \dfrac{}{\vdash C^{\perp}, C}}{\vdash B, \boxed{((A \otimes B^{\perp}) \parr A^{\perp})}}}{\vdash C^{\perp}, B, \boxed{(C \otimes ((A \otimes B^{\perp}) \parr A^{\perp}))}}}{\vdash \boxed{(C^{\perp} \parr B)}, (C \otimes ((A \otimes B^{\perp}) \parr A^{\perp}))} \qquad (1)$$

This derivation may be transformed into the following proof-net:



(2)

More abstractly, the above proof-net may be identified with the following graph:



(3)

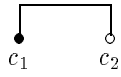
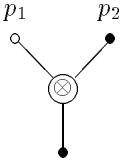
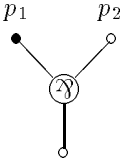
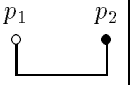
which represents the core of Derivation (1).

On the formal side, Girard defines first the notion of proof structure, which corresponds to a class of graphs akin to Graph 3. Then, a global geometrical criterion allows the proof-nets, which are the graphs that correspond actually to sequential proofs, to be discriminated from the other proof-structures (see [3, 6, 10, 20] for instances of such criteria and for more details).

2.2 Intuitionistic Multiplicative Proof-Nets

The examples in the previous section are taken from the so-called classical multiplicative linear logic. Now, logics such as the Lambek calculus are intuitionistic in the technical sense that they are defined by means of sequent calculi whose sequents are made of several antecedent formulas and only one succedent formula. In order to accommodate the notion of proof-net to such logics, one must use a notion of input ($\bullet$) and output ($\circ$) polarities [2, 10]. The idea is that the input (or negative) polarities correspond to those occurrences of formulas that appear in the antecedents of the sequents while the output (or positive) polarities correspond to the occurrences of formulas that appear in the succedents.

More precisely, consider the following table that defines the notion of links for the proof-structures of the intuitionistic implicative linear logic (IILL, also known as van Benthem's LP*), whose only connective is the linear implication $\multimap$.

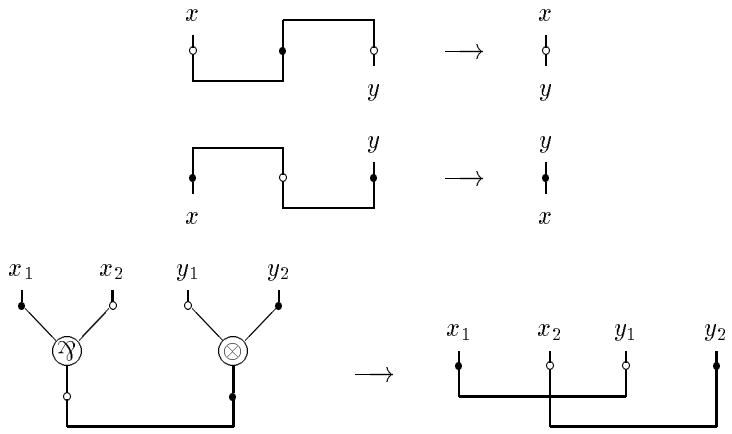
NAME	*Axiom*	*Tensor*	*Par*	*Cut*
LINK				
PREMISES	none	p_1, p_2	p_1, p_2	p_1, p_2
CONCLUSIONS	c_1, c_2	c	c	none
TYPES	$c_1 : A^-$ $c_2 : A^+$	$p_1 : A^+$ $p_2 : B^-$ $c : (A \multimap B)^-$	$p_1 : A^-$ $p_2 : B^+$ $c : (A \multimap B)^+$	$p_1 : A^+$ $p_2 : A^-$
POLARITIES	c_1 : input c_2 : output	p_2, c : input p_1 : output	p_1 : input p_2, c : output	p_2 : input p_1 : output

Proof-structures are then defined to be graphs made of 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.

Then, as we have already pointed out, a correctness criterion allows one to distinguish the proof-nets among the proof-structures. In fact, this correctness criterion ensures that the proof-nets are those proof-structures that may be sequentialized into Gentzen-like derivations. In particular, it ensures that any proof-net has exactly one output conclusion.

The proof-net formalism also captures the dynamics of proofs: cut elimination may be performed directly on the proof-nets without any reference to the sequent calculus. Moreover, the cut elimination process is specified by simple graph rewriting rules. In the case of ILLL, these rewriting rules, which are purely local, are the following:



2.3 Encoding of the Linear λ-Terms into the Intuitionistic Multiplicative Proof-Nets

The proof-nets introduced in the previous section give a way of representing the proofs of intuitionistic implicative linear logic. Another way of representing these proofs is given by the Curry-Howard isomorphism. It consists of using the so-called linear λ-terms. This raises immediately the following question: what is the relationship between these two different formalisms? The answer is quite simple: the linear λ-terms may be encoded as proof-nets and, in the case of linear λ-terms in normal form, the resulting correspondence is one-one.

As is well known, λ-terms in normal form may be defined by the following grammar:

$$\begin{array}{rcl} \mathcal{A} & ::= & x \mid (\mathcal{A}\,\mathcal{T}) \\ \mathcal{T} & ::= & \mathcal{A} \mid \lambda x.\mathcal{T} \end{array}$$

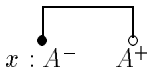
By adding the constraint that each λ must bind exactly one variable occurrence, one defines the linear λ-terms that correspond, through the Curry-Howard isomorphism, to the proofs of IILL.

Now, the encoding of these terms into proof-nets obeys the following principles:

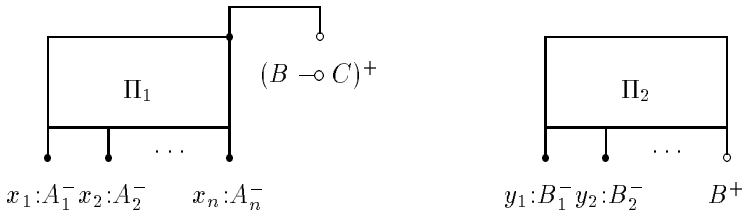
1. to any λ-term of type A with free variables x_i of types A_i corresponds some proof-net whose unique output conclusion is of type A and whose input conclusions, which are of types A_i, may be labelled with the variables x_i.
2. to any λ-term defined by the non-terminal $\mathcal{A}$ corresponds some proof-net whose unique output conclusion is the output conclusion of an *axiom*-link.

Keeping these two invariants in mind, one may define the encoding by induction on the above grammar.

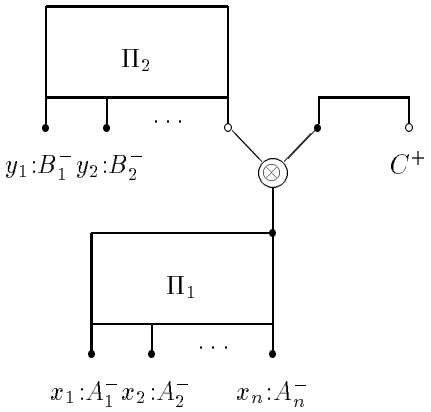
Case 1: variable. The proof-net encoding a variable x is made of an *axiom*-link:



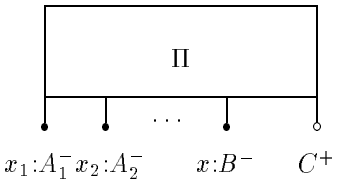
Case 2: application. Let Π_1 and Π_2 be the two proof-nets encoding respectively $\mathcal{A}$ and $\mathcal{T}$:



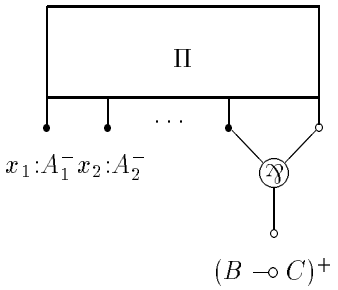
The proof-net encoding $(\mathcal{A}\,\mathcal{T})$ is obtained by gluing Π_1 and Π_2 with a *tensor*-link as follows:



Case 3: abstraction. Let Π be the proof-net encoding $\mathcal{T}$:

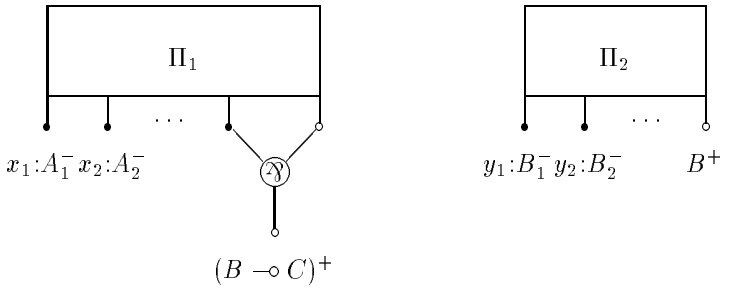


The proof-net encoding $\lambda x.\mathcal{T}$ is obtained by adding a *par*-link as follows:

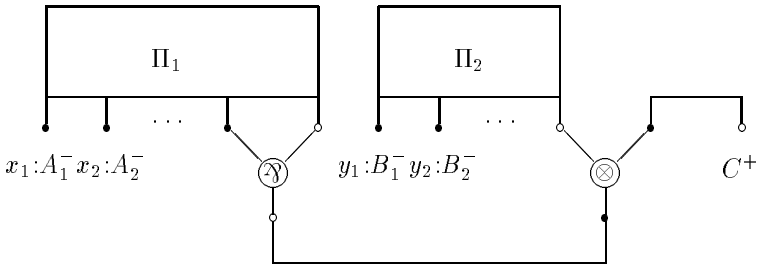


The above encoding is not merely syntactic: it relies on an actual correspondence that also takes into account the dynamics of the λ-calculus. Indeed, β-redexes may be represented by using *cuts* as follows.

Let Π_1 and Π_2 be the proof-nets encoding respectively $\lambda x.\mathcal{T}_1$ and T_2:



The proof-net encoding $((\lambda x.\mathcal{T}_1)\, \mathcal{T}_2)$ is obtained as follows:



Then, the process of cut elimination, as specified at the end of Section 2.2, amounts to the reduction of the β-redexes.

3 Semantic Readings as Proof-Nets

As noticed by van Benthem, λ-terms provide a semantic reading of categorial laws such as Montague type raising, Geach composition, argument lowering, etc.

Consider, for instance, Montague type raising, i.e.,

$$e \vdash ((e,t),t)$$

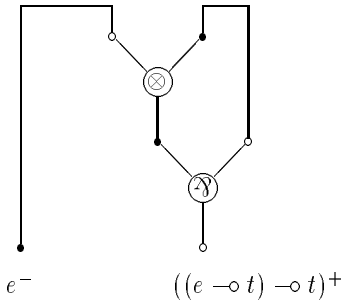
or, using Girard's notation,

$$e \vdash ((e \multimap t) \multimap t).$$

Its semantic reading is provided by the λ-term $\lambda x.x\, y$, where y is a free variable of type e. In the case of a sequential proof, it is necessary to decorate each sequent with a λ-term in order to get this semantic reading:

$$\dfrac{\dfrac{x : (e \multimap t) \vdash x : (e \multimap t) \qquad y : e \vdash y : e}{y : e,\, x : (e \multimap t) \vdash x\, y : t}}{y : e \vdash \lambda x.\, x\, y : ((e \multimap t) \multimap t)}$$

When using proof-nets, however, one may get the semantic reading directly. Consider the proof-net that proves Montague type raising:



As we will see, a simple traversal of this proof-net will provide the semantic reading. This traversal, which follows Lamarche's dependency paths [10], may be specified by a simple set of instructions:

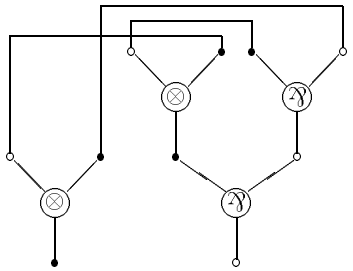
1. enter the proof-net by its unique output conclusion;
2. follow the path specified by the output polarities until an *axiom*-link is eventually reached; this path, which is ascending, is made of *par*-links that correspond to successive λ-abstractions;
3. cross the *axiom*-link following the output-input direction;
4. follow the path specified by the input polarities; this path, which is descending, is made of *tensor*-links that correspond to successive applications; it ends either on some input conclusion of the proof-net, or on the input premise of some *par*-link; in both cases, the end of the path coincides with the head-variable of the corresponding λ-term; in the first case (input conclusion), this head-variable is free; in the second case (premise of a *par*-link) this head-variable is bound to the λ corresponding to the *par*-link;
5. in order to get all the arguments to which the head-variable is applied, start again the same sort of traversal from every output premise of the *tensor*-links that have been encountered during the descending phase described in 4;

It is worth noting that the above traversal algorithm does not make sense on every proof-structure. For instance, one may easily imagine proof-structures some links of which would never have been visited during the traversal. Another possible problem is when reading the head-variable: the descending path that follows the input polarities could end on the input premise of a *par*-link that would not have been visited before, i.e., a *par*-link that would not correspond to a λ-abstraction. But these "pathological" proof-structures, for which the reading algorithm does not work, are precisely the ones that are rejected by the correctness criterion. In other words, they are not proof-nets.

As a further illustration, consider the following consequence of Geach composition rule:

$$(a \multimap b) \vdash (c \multimap a) \multimap (c \multimap b),$$

to which is associated the following proof-net:



$(a \multimap b)^-$ $\quad$ $((c \multimap a) \multimap (c \multimap b))^+$

Let us try to apply the reading algorithm on this example:

1. we enter the proof-net by its output conclusion (i.e. the conclusion of type $((c \multimap a) \multimap (c \multimap b))^+$), and go up, following the output polarities; we cross two *par*-links that correspond to two successive λ-abstractions, say λx and λy; hence, the λ-term that we are reading has the form $\lambda x.\lambda y.\mathcal{T}_1$
2. we follow the *axiom*-link in the output-input direction, and go down, following the intput polarities; we cross the leftmost *tensor*-link, and we end on the input conclusion of type $(a \multimap b)^-$; this input conclusion corresponds to a free head-variable, say z; therefore, we are reading a λ-term of the form $\lambda x.\lambda y.(z\ \mathcal{T}_2)$;

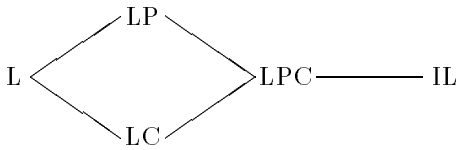
3. in order to read the argument to which is applied the head-variable z (i.e. the λ-term $\mathcal{T}_2$), we again start the process from the output conclusion of the *tensor*-link that we just crossed; we follow the leftmost *axiom*-link, cross the second *tensor*-link and we end on the input premise of the *par*-link corresponding to λx; hence, we have read a λ-term of the form $\lambda x.\lambda y.(z\ (x\ \mathcal{T}_3))$;
4. similarly, we read the λ-term corresponding to $\mathcal{T}_3$ and we get the complete reading of the proof-net: $\lambda x.\lambda y.(z\ (x\ y))$.

Thus, we have shown how to obtain semantic readings from proof-nets by a simple traversal following Lamarche's dependency paths. In fact, this traversal of the proof-nets is so simple (linear time) that one can say that the semantic reading is no longer provided by a λ-term but by the proof-net itself. In other words, we argue that we no longer need the λ-terms anymore since we have the proof-nets at our disposal. This point of view will make more sense when working with logics more powerfull than IILL in which the correspondence between λ-terms and proof-nets is no longer one-one. Indeed, for such logics, the notion of proof-net is much richer than that of λ-term.

4 Proof-Nets for the Categorial Hierarchy

4.1 The Categorial Hierarchy

In [24], van Benthem defines the following categorial hierarchy, starting on the left with the Lambek calculus and ending on the right with the intuitionistic implicative logic:



Each of these implicational caculi may be obtained from another by adding or removing one or more structural rules. For instance, starting from L, one gets LC by admitting the *contraction*-rule. Then, LPC is obtained by adding the *permutation*-rule. Finally, one reaches IL by adding to LPC by the *weakening*-rule.

In Section 2, we have introduced the notion of proof-net in the framework of IILL, which is another name for LP.[1] Therefore, in order to adapt this notion to the Lambek calculus, we must explain how to reject the *permutation*-rule. This will be explained briefly in Section 4.2.

On the other hand, in order to adapt the notion of proof-net to LC, LPC, and IL, we must allow for the structural rules of contraction and weakening. This is done, in linear logic, by using Girard's modal operator "!". Hence, we will consider, in Section 4.3, a the fragment of intuitionistic linear logic that contains "$\multimap$" and "!" as the only connectives. This fragment is called intuitionistic implicative exponential linear logic (IIELL, for short).

4.2 Proof-Nets for the Lambek calculus

Proof-nets for the lambek calculus have been defined by Roorda in his thesis [21] and are presented in detail in [11, 20].

In order to deal with the non-commutativity of L, one must distinguish between the left and right premises of the links. Consequently, one gets two different sorts of *tensor*-links, corresponding to the formulas $(A\backslash B)^-$ and $(A/B)^-$, and two different sorts of *par*-links, corresponding to the formulas $(A\backslash B)^+$ and $(A/B)^+$. One must also take into account the fact that the formulas in Lambek's sequents are ordered. This gives rise to an order on the

[1] There is actually one difference: in LP, the empty antecedent is not admitted in the sequents.

conclusions of the proof-nets (a cyclical order, to be precise). Then, in order to adapt the correctness criterion, one has to add a planarity requirement: *the axiom-links may not cross one another.*

4.3 Proof-nets for Intuitionistic Implicative Exponential Linear Logic

Girard's unary connective "!" is a modal operator that allows for the structural rules of contraction and weakening. It obeys the following logical rules:

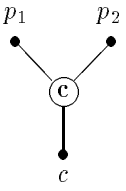
$$\frac{A, \Gamma \vdash C}{!A, \Gamma \vdash C} \quad \text{(dereliction)} \qquad \frac{!A_1, \ldots !A_n \vdash C}{!A_1, \ldots !A_n \vdash !C} \quad \text{(promotion)}$$

$$\frac{!A, !A, \Gamma \vdash C}{!A, \Gamma \vdash C} \quad \text{(contraction)} \qquad \frac{\Gamma \vdash C}{!A, \Gamma \vdash C} \quad \text{(weakening)}$$

Consequently, intuitionisitc implication "$\rightarrow$" may be defined as follows:

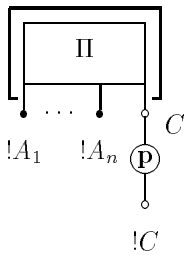
$$(A \rightarrow B) \;=\; (!A \multimap B).$$

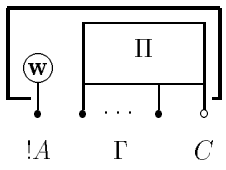
Now, in order to accomodate the proof-nets with this modal operator, new sorts of links must be introduced:

NAME	*Dereliction*	*Contraction*
LINK		
PREMISES	p	p_1, p_2
CONCLUSIONS	c	c
TYPES	$p : A^-$ $c : !A^-$	$p_1, p_2, c : !A^-$
POLARITIES	p, c : input	p_1, p_2, c : input

As for *promotion* and *weakening*, simple links are not sufficient. The problem with the promotion rule is that it is contextual:

In order to circumvent these difficulties, one uses *boxes*: A box is a part of the proof structure, the interior of which is itself a proof structure. The conclusions of a box are called its doors. There are two kinds of boxes: *promotion*-boxes and *weakening*-boxes.

promotion-box : 

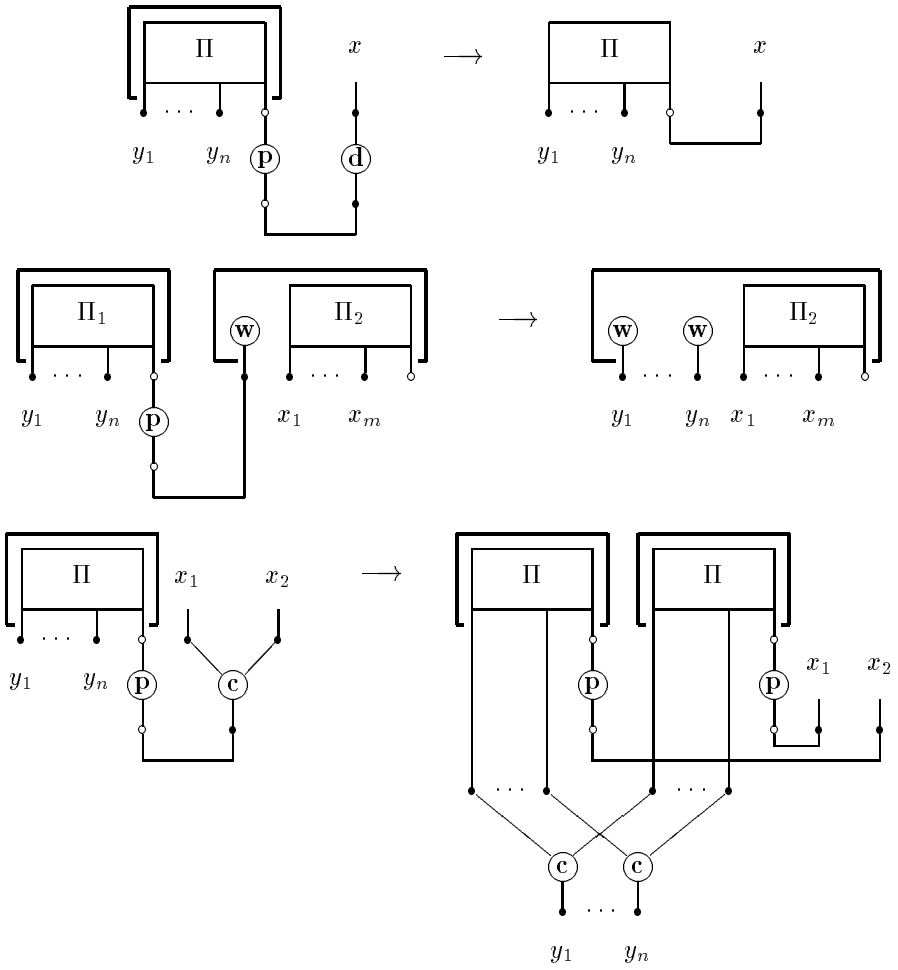
weakening-box : 

Then, the notion of correctness is defined by induction on the nesting of the boxes:

1. the proof-net obtained by replacing each box by a n-ary *axiom*-link whose conclusions are the doors of the boxes satisfies the usual correctness criterion;

2. the interior of each box is correct.

On the dynamic side, the presence of the connective "!" gives rise to new cut-elimination cases that correspond to new graph rewriting rules:



where, strictly speaking, the n-ary *weakening*-box in the lefthand side of the second rule correspond to n nested ordinary *weakening*-boxes.

4.4 Encoding simply-typed λ-terms

The intuitionistic implicative logic corresponds, through the Curry-Howard isomorphism, to the simply-typed λ-calculus. Hence, the simply-typed λ-terms may be encoded into the IIELL proof-nets.

The translation is very similar to the one described in section 2. The main difference is that a λ-term whose free variables are of type A_i is now translated into a proof-net whose input conclusions are of type $!A_i$. This allows different input conclusions of the same type to be contracted, which is a way of taking the non-linearity of the terms into account. Then, the translation has to be adapted by using *contraction*-links, *dereliction*-links, *weakening*-boxes, and *promotion*-boxes when needed.

Conversely, the traversal algorithm that provides the λ-term corresponding to a proof-net is almost the same as the one described in Section 3. The only difference is during the

descending phase: after having crossed a bunch of *tensor*-links, one may cross a *dereliction*-link followed by several *contraction*-links before reaching the head-variable.

For more details on this topic, see [2, 4, 19].

5 Semantic Recipes as Proof-Nets

In this section we provide an example of the use of proof-nets as a uniform framework that allows syntactic and semantic analysis to be unified. This example, which is more elaborate that those of Section 3, is in the spirit of Montague PTQ Grammar[14].

On the syntactic side, we use the Lambek calculus together with three basic types: n, sn, and s.

On the semantic side, we use IIELL with the two Montagovian basic types e and t. For the purpose of our example, we also assume the existence of the following typed constants (with their obvious intended meanings):

$$\forall : ((e \multimap t) \multimap t), \quad \supset : (t \multimap (t \multimap t)), \quad \mathsf{barber} : (e \multimap t), \quad \mathsf{shave} : (e \multimap (e \multimap t)).$$

Technically, these constants will be handled as free variables: they will decorate input conclusions of proof-nets.

We also assume that the following homomorpism $\mathcal{H}$ between syntactic and semantic types is given:

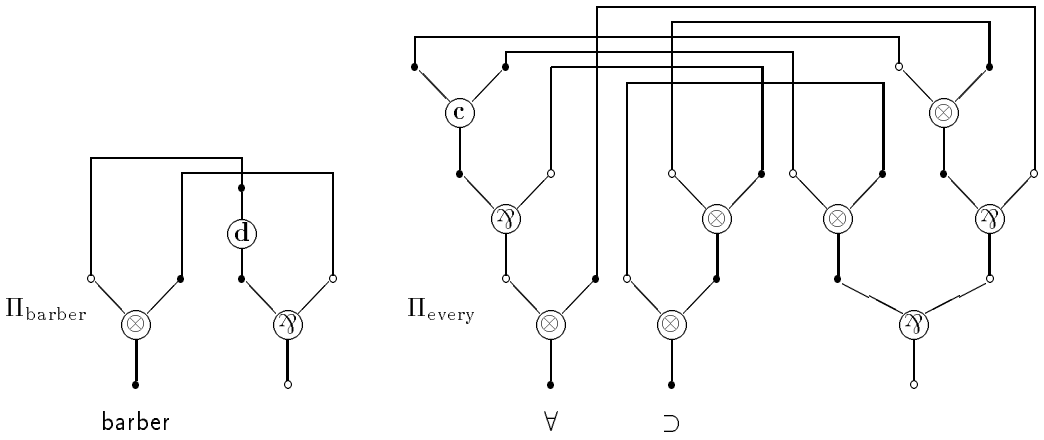
$$\mathcal{H}(n) = (!e \multimap t) \qquad \mathcal{H}(sn) = !e \qquad \mathcal{H}(s) = t$$
$$\mathcal{H}(A\backslash B) = \mathcal{H}(A) \multimap \mathcal{H}(B) \qquad \mathcal{H}(A/B) = \mathcal{H}(B) \multimap \mathcal{H}(A)$$

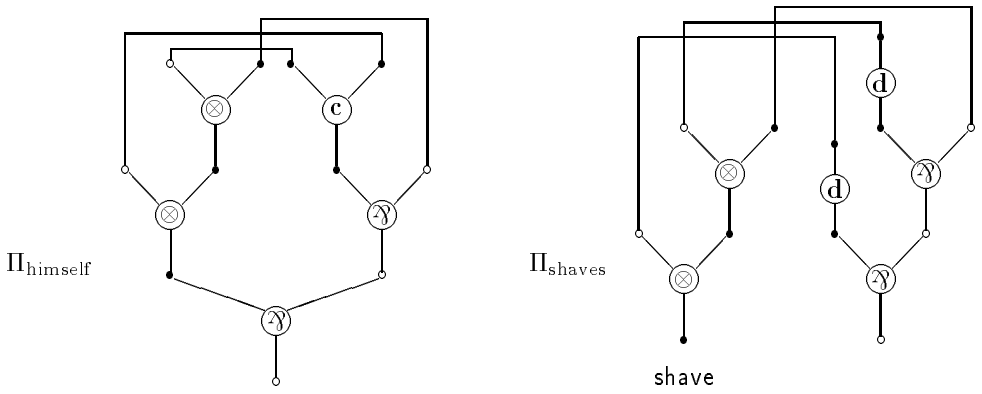
Then, we consider a lexicon made of words to which syntactic types and semantic proof-nets are attached. Each of these semantic proof-nets is such that:

1. the type of its unique output conclusion is the semantic type of the corresponding word, i.e., the homomorphic image of the the associated syntactic type;
2. its input conclusions (if any) are decorated with constants.

LEXICON			
WORD	SYNTACTIC CATEGORY	SEMANTIC CATEGORY	SEMANTIC PROOF-NET
barber	n	$(!e \multimap t)$	Π_{barber}
every	$(s/(np\backslash s))/n$	$(!e \multimap t) \multimap ((!e \multimap t) \multimap t)$	Π_{every}
himself	$((np\backslash s)/np)\backslash(np\backslash s)$	$(!e \multimap (!e \multimap t)) \multimap (!e \multimap t)$	Π_{himself}
shaves	$np\backslash(s/np)$	$(!e \multimap (!e \multimap t))$	Π_{shaves}

where the proof-nets Π_{barber}, Π_{every}, Π_{himself}, Π_{shaves} are respectively the following:





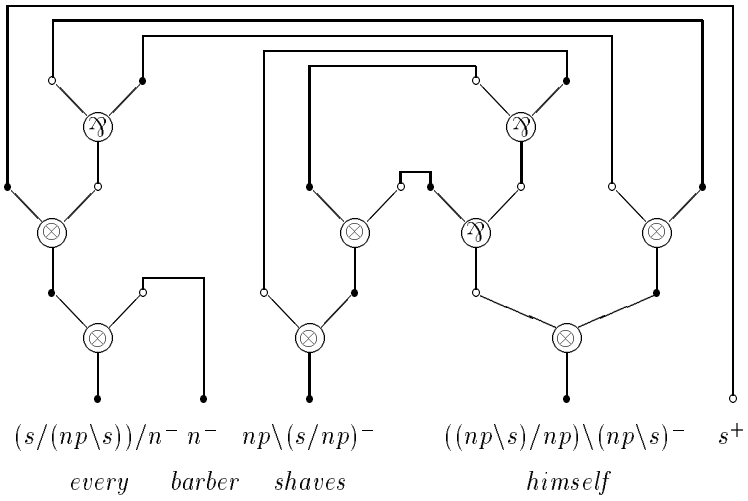
These semantic proof-nets correspond to semantic recipes. For instance, by applying the reading algorithm of Sections 3 and 4 on Π_{every}, one obtains the following λ-term:

$$\lambda P.\lambda Q.\forall\,(\lambda x.\supset (P\,x)\,(Q\,x))$$

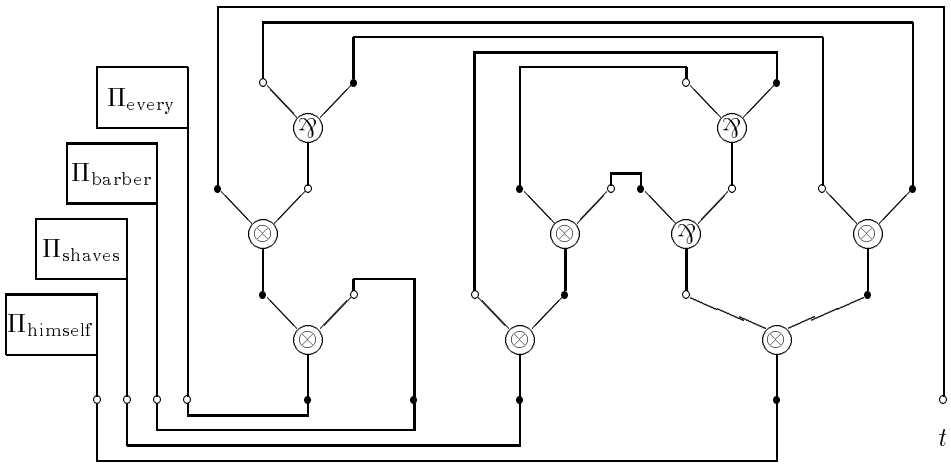
or, using a more usual notation for the connectives $\forall$ and $\supset$:

$$\lambda P.\lambda Q.\forall x.(P\,x)\ \supset\ (Q\,x)$$

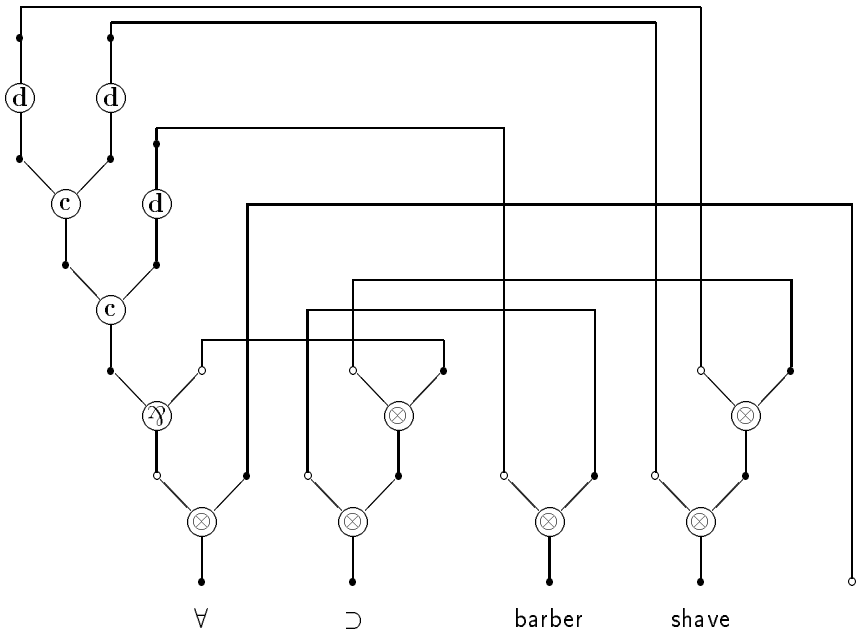
Now, consider the non-commutative proof-net resulting from the syntactic analysis of the sentence *"every barber shaves himself"*:



If one does not take the orientation of the links into account (identifying both $\backslash$ and $/$ with $\multimap$), we may think of the above proof-net as a proof-net of IILL or even IIELL. Then, replacing each syntactic type by the corresponding semantic type does not affect the well-typedness of the proof-net because $\mathcal{H}$ is a homomorphism. This allows the semantic recipes to be plugged by means of cuts, the resulting proof-net being still well-typed:



Then, by cut elimination, one gets the semantic proof-net associated to the whole sentence:



whose semantic reading provides: $\forall x.(\mathsf{barber}\, x) \supset (\mathsf{shave}\, \mathsf{x}\, \mathsf{x})$.

Acknowlegment

We would like to thank A. Cichon and F. Lamarche for helpful comments.

References

[1] B. Carpenter. *Lectures on Type-Logical Semantics.* MIT Press, Cambridge, Massachussetts and London England, 1996.

[2] V. Danos. *Une application de la logique linéaire à l'étude des processus de normalisation et principalement du lambda calcul.* Thèse de doctorat, Université de Paris VII, 1990.

[3] V. Danos and L. Regnier. The structure of multiplicatives. *Archive for Mathematical Logic*, 28:181–203, 1989.

[4] V. Danos and L. Reigner. Local and asynchronous beta-reductions. In *Proceedings of the eighth annual symposium on Logic in Computer Science*, 1993.

[5] Ph. de Groote, editor. *The Curry-Howard Isomorphism*, volume 8 of *Cahier du centre de Logique*. Université Catholique de Louvain, Academia, 1995.

[6] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[7] J.-Y. Girard. Linear logic: its syntax and semantics. In J.-Y. Girard, Y. Lafont, and L. Regnier, editors, *Advances in Linear Logic*, London Mathematical Society Lecture Notes. Cambridge University Press, 1995.

[8] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*, volume 7 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1989.

[9] W.A. Howard. The formulae-as-types notion of construction. In J. P. Seldin and J. R. Hindley, editors, *to H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 479–490. Academic Press, 1980.

[10] F. Lamarche. Games semantics for full propositional linear logic. In *Ninth Annual IEEE Symposium on Logic in Computer Science*. IEEE Press, 1995.

[11] F. Lamarche and C. Retoré. Proof nets for the lambek calculus. In M. Abrusci, C. Casadio, and G. Sandri, editors, *Third Roma Workshop: Proofs and Linguistic Categories*, Rapporto di Ricerca del Dipartimento de Filosofia. Università di Bologna, 1996.

[12] J. Lambek. The mathematics of sentence structure. *Amer. Math. Monthly*, 65:154–170, 1958.

[13] A. Lecomte. Towards efficient parsing with proofnets. In *11*$^{\text{th}}$ *Conference of European chapter of the Association for Computational Linguistics*. Utrecht, 1993.

[14] R. Montague. The proper treatment of quantification in ordinary english. In J. Hintikka, J. Moravcsik, and P. Suppes, editors, *Approaches to natural language: proceedings of the 1970 Stanford workshop on Grammar and Semantics*, Dordrecht, 1973. Reidel.

[15] M. Moortgat. *Categorial Investigations: logical and linguistic aspects of the lambek calculus*. Foris Publications, 1988.

[16] G. Morrill. Memoisation of categorial proof nets: parallelism in categorial processing. Technical Report LSI-96-24-R, Dept. de Llengatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, 1996.

[17] G.V. Morrill. *Type Logical Grammar*. Kluwer Academic Publishers, Dordrecht and Hingham, 1994.

[18] A. Ranta. *Type theoretical grammar*. Oxford University Press, 1994.

[19] L. Regnier. *Lambda calcul et Réseaux*. Thèse de doctorat, spécialité mathématiques, Université Paris 7, Janvier 1992.

[20] C. Retoré. Des réseaux de démonstration pour la linguistique: une introduction à la logique linéaire. *Traitement Automatique de Langues*, 1996. à paraître.

[21] D. Roorda. *Resource Logics: proof-theoretical investigations*. PhD thesis, University of Amsterdam, 1991.

[22] A. Troelstra. *Lectures on Linear Logic*, volume 29 of *CSLI Lecture Notes*. Center for the Study of Language and Information, 1992.

[23] J. van Benthem. The semantics of variety in categorial grammar. In W. Buszkowski, W. Marciszewski, and J. van Benthem, editors, *Categorial Grammars*. John Benjamins, 1988.

[24] J. van Benthem. *Language in action: Categories, Lambdas and Dynamic Logic*, volume 130 of *Sudies in Logic and the foundation of mathematics*. North-Holland, Amsterdam, 1991.